



# **BIOS and Kernel Developer's Guide for AMD Athlon™ 64 and AMD Opteron™ Processors**



Publication # <b>26094</b>	Revision: <b>3.28</b>
Issue Date: <b>October 2005</b>	

© 2002, 2003, 2004, 2005 Advanced Micro Devices, Inc. All rights reserved.

The contents of this document are provided in connection with Advanced Micro Devices, Inc. ("AMD") products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD's Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

## **Trademarks**

AMD, the AMD Arrow logo, AMD Athlon, AMD Opteron, and combinations thereof, 3DNow!, AMD PowerNow!, and Cool'n'Quiet are trademarks of Advanced Micro Devices, Inc.

HyperTransport is a licensed trademark of the HyperTransport Technology Consortium.

MMX is a trademark and Pentium is a registered trademark of Intel Corporation.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

# Contents

---

	<b>Revision History .....</b>	<b>19</b>
<b>1</b>	<b>Introduction and Overview .....</b>	<b>27</b>
1.1	About This Guide .....	27
1.2	Related Documents .....	28
1.3	Conventions and Definitions .....	28
1.3.1	Notation .....	28
1.4	Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors .....	29
<b>2</b>	<b>Processor Initialization and Configuration .....</b>	<b>33</b>
2.1	Bootstrap Processor Initialization .....	33
2.1.1	Detecting AP and Initializing Routing Table .....	33
2.1.2	Initializing Noncoherent HyperTransport™ Technology Devices .....	34
2.1.3	Initializing Link Width and Frequency .....	34
2.1.4	Initializing the Memory Controller on All Processor Nodes .....	35
2.1.5	Initializing the Address Map Table .....	35
2.2	Application Processor Initialization .....	35
2.3	BIOS Requirement for 64-Bit Operation .....	36
2.3.1	Sizing and Testing Memory above 4 Gbytes .....	36
2.3.2	Using BIOS Callbacks in 64-Bit Mode .....	36
<b>3</b>	<b>Memory System Configuration .....</b>	<b>37</b>
3.1	Configuration Space Accesses .....	37
3.1.1	Configuration Address Register .....	37
3.1.2	Configuration Data Register .....	38
3.2	Memory System Configuration Registers .....	39
3.3	Function 0—HyperTransport™ Technology Configuration .....	39
3.3.1	Device/Vendor ID Register .....	41
3.3.2	Status/Command Register .....	42

3.3.3	Class Code/Revision ID Register .....	42
3.3.4	Header Type Register .....	43
3.3.5	Capabilities Pointer Register .....	43
3.3.6	Routing Table Node <i>i</i> Registers .....	44
3.3.7	Node ID Register .....	46
3.3.8	Unit ID Register .....	47
3.3.9	HyperTransport™ Transaction Control Register .....	48
3.3.10	HyperTransport™ Initialization Control Register .....	53
3.3.11	LDTi Capabilities Register .....	54
3.3.12	LDTi Link Control Registers .....	56
3.3.13	LDTi Frequency/Revision Registers .....	60
3.3.14	LDTi Feature Capability Registers .....	61
3.3.15	LDTi Buffer Count Registers .....	62
3.3.16	LDTi Bus Number Registers .....	64
3.3.17	LDTi Type Registers .....	65
3.4	Function 1—Address Map .....	66
3.4.1	Device/Vendor ID Register .....	68
3.4.2	Class Code/Revision ID Register .....	69
3.4.3	Header Type Register .....	69
3.4.4	DRAM Address Map .....	70
3.4.5	Memory-Mapped I/O Address Map Registers .....	73
3.4.6	PCI I/O Address Map Registers .....	76
3.4.7	Configuration Map Registers .....	78
3.4.8	DRAM Hole Address Register .....	80
3.5	Function 2—DRAM Controller .....	80
3.5.1	Device/Vendor ID Register .....	82
3.5.2	Class Code/Revision ID Register .....	83
3.5.3	Header Type Register .....	83
3.5.4	DRAM CS Base Address Registers .....	84
3.5.5	DRAM CS Mask Registers .....	87

3.5.6	DRAM Bank Address Mapping Register .....	88
3.5.7	Address to Node/Chip Select Translation .....	98
3.5.8	Memory Hoisting .....	99
3.5.9	DRAM Timing Low Register .....	102
3.5.10	DRAM Timing High Register .....	104
3.5.11	DRAM Configuration Registers .....	106
3.5.12	DRAM Delay Line Register .....	115
3.5.13	Scratch Register .....	117
3.6	Function 3—Miscellaneous Control .....	117
3.6.1	Device/Vendor ID Register .....	119
3.6.2	Class Code/Revision ID Register .....	120
3.6.3	Header Type Register .....	120
3.6.4	Machine Check Architecture (MCA) Registers .....	121
3.6.5	ECC and Chip Kill Error Checking and Correction .....	137
3.6.6	Scrub Control Register .....	140
3.6.7	DRAM Scrub Address Registers .....	141
3.6.8	XBAR Flow Control Buffers .....	143
3.6.9	XBAR-to-SRI Buffer Count Register .....	150
3.6.10	Display Refresh Flow Control Buffers .....	151
3.6.11	Power Management Control Registers .....	151
3.6.12	GART Aperture Control Register .....	154
3.6.13	GART Aperture Base Register .....	155
3.6.14	GART Table Base Register .....	156
3.6.15	GART Cache Control Register .....	157
3.6.16	Clock Power/Timing Low Register .....	157
3.6.17	Clock Power/Timing High Register .....	160
3.6.18	HyperTransport™ FIFO Read Pointer Optimization Register .....	161
3.6.19	Thermtrip Status Register .....	163
3.6.20	Northbridge Capabilities Register .....	165
<b>4</b>	<b>DRAM Configuration .....</b>	<b>169</b>

4.1	Programming Interface .....	169
4.1.1	SPD ROM-Based Configuration .....	169
4.1.2	Non-SPD ROM-Based Configuration .....	172
4.1.3	Maximum DRAM Speed as a Function of Loading .....	175
4.1.4	DIMM Matching Algorithm .....	184
4.1.5	DRAM Initialization .....	185
4.2	DRAM Configuration .....	186
<b>5</b>	<b>Machine Check Architecture .....</b>	<b>189</b>
5.1	Determining Machine Check Support .....	189
5.2	Machine Check Errors .....	189
5.2.1	Sources of Machine Check Errors .....	190
5.3	Machine Check Architecture Registers .....	192
5.3.1	Global Machine Check Model-Specific Registers (MSRs) .....	192
5.3.2	Error Reporting Bank Machine Check MSRs .....	195
5.4	Error Reporting Banks .....	199
5.4.1	Data Cache (DC) .....	199
5.4.2	Instruction Cache (IC) .....	204
5.4.3	Bus Unit (BU) .....	207
5.4.4	Load Store Unit (LS) .....	212
5.4.5	Northbridge (NB) .....	213
5.5	Initializing the Machine Check Mechanism .....	214
5.6	Using Machine Check Features .....	214
5.6.1	Handling Machine Check Exceptions .....	215
<b>6</b>	<b>System Management Mode (SMM) .....</b>	<b>219</b>
6.1	SMM Overview .....	219
6.2	Operating Mode and Default Register Values .....	219
6.3	SMM State Save Definition .....	221
6.4	SMM Initial State .....	223
6.5	SMM-Revision Identifier .....	224
6.6	SMM Base Address .....	225

6.7	Auto Halt Restart .....	225
6.8	SMM I/O Trap and I/O Restart .....	226
6.8.1	SMM I/O Trap .....	227
6.8.2	SMM I/O Restart Byte .....	227
6.9	Exceptions and Interrupts in SMM .....	228
6.10	NMI Mask .....	228
6.11	Protected SMM and ASeg/TSeg .....	229
6.11.1	SMM_MASK Register .....	229
6.11.2	SMM_ADDR Register .....	230
6.11.3	SMM ASeg .....	231
6.11.4	SMM TSeg .....	231
6.11.5	Closing SMM .....	232
6.11.6	Locking SMM .....	232
6.12	SMM Special Cycles .....	233
<b>7</b>	<b>HyperTransport™ Technology Configuration and Enumeration .....</b>	<b>235</b>
7.1	Initial Configuration Steps .....	235
7.2	One-Node Coherent HyperTransport™ Technology Initialization .....	236
7.3	Two-Node Coherent HyperTransport™ Technology Initialization .....	236
7.4	Generic HyperTransport™ Technology Configuration .....	236
7.5	Rules for Valid Routing Tables .....	237
7.5.1	Two-Hop Cycle Example .....	237
<b>8</b>	<b>Advanced Programmable Interrupt Controller (APIC) .....</b>	<b>239</b>
8.1	Interrupt Delivery .....	240
8.2	Vectored Interrupt Handling .....	240
8.3	Spurious Interrupts .....	241
8.3.1	Spurious Interrupts Caused by Timer Tick Interrupt .....	241
8.4	Lowest-Priority Arbitration .....	242
8.5	Inter-Processor Interrupts .....	243
8.6	APIC Timer Operation .....	243
8.7	State at Reset .....	243

8.8	Register Summary .....	244
8.8.1	APIC ID Register .....	245
8.8.2	APIC Version Register .....	245
8.8.3	Task Priority Register .....	246
8.8.4	Arbitration Priority Register .....	246
8.8.5	Processor Priority Register .....	247
8.8.6	End of Interrupt Register .....	247
8.8.7	Logical Destination Register .....	247
8.8.8	Destination Format Register .....	248
8.8.9	Spurious Interrupt Vector Register .....	248
8.8.10	In-Service Registers .....	249
8.8.11	Trigger Mode Registers .....	250
8.8.12	Interrupt Request Registers .....	251
8.8.13	Error Status Register .....	252
8.8.14	Interrupt Command Register Low .....	253
8.8.15	Interrupt Command Register High .....	254
8.8.16	Timer Local Vector Table Entry .....	255
8.8.17	Performance Counter Local Vector Table Entry .....	256
8.8.18	Local Interrupt 0 (Legacy INTR) Local Vector Table Entry Register .....	256
8.8.19	Local Interrupt 1 (Legacy NMI) Local Vector Table Entry .....	257
8.8.20	Error Local Vector Table Entry .....	258
8.8.21	Timer Initial Count Register .....	258
8.8.22	Timer Current Count Register .....	259
8.8.23	Timer Divide Configuration Register .....	259
<b>9</b>	<b>Power and Thermal Management .....</b>	<b>261</b>
9.1	Stop Grant .....	262
9.2	C-States .....	264
9.2.1	C1 Halt State .....	264
9.2.2	C2 and C3 .....	264
9.2.3	C3 and AltVID .....	264



9.3	Throttling .....	265
9.4	Processor ACPI Thermal Zone .....	265
9.5	Processor Performance States .....	265
9.5.1	BIOS Requirements for P-State Transitions .....	266
9.5.2	BIOS Requirements for P-State Transitions in Systems Using Dual Core Processors .....	268
9.5.3	BIOS-Initiated P-State Transitions .....	268
9.5.4	BIOS Support for Operating System/CPU Driver-Initiated P-State Transitions ..	268
9.5.5	Processor Driver Requirements .....	269
9.5.6	P-State Transition Sequence .....	269
9.6	ACPI 2.0 Processor P-State Objects .....	277
9.6.1	_PCT (Performance Control) .....	278
9.6.2	_PSS (Performance-Supported States) .....	278
9.6.3	_PPC (Performance Present Capabilities) .....	284
9.6.4	PSTATE_CNT .....	286
9.6.5	CST_CNT .....	286
9.7	BIOS Support for AMD PowerNow!™ Software with Legacy Operating Systems ..	287
9.8	System Configuration for Power Management .....	288
9.8.1	Chipset Configuration for Power Management .....	288
9.8.2	Processor Configuration for Power Management .....	289
<b>10</b>	<b>Performance Monitoring .....</b>	<b>295</b>
10.1	Performance Counters .....	295
10.2	Performance Event-Select Registers .....	296
10.2.1	Performance Monitor Events .....	298
<b>11</b>	<b>CPUID .....</b>	<b>319</b>
<b>12</b>	<b>BIOS Checklist .....</b>	<b>327</b>
12.1	CPUID .....	327
12.2	CPU Speed Detection .....	327
12.3	HyperTransport™ Link Detection .....	327
12.4	HyperTransport™ Link Frequency Selection .....	328

12.5	Multiprocessing Capability Detection .....	328
12.6	Setup for Dual Core Processors .....	329
12.7	APIC ID Assignment Requirements .....	330
12.8	Processors With Different Revisions or Maximum Frequencies in Multiprocessor Systems .....	331
12.9	Model-Specific Registers (MSRs) .....	332
12.9.1	Software Considerations for Accessing Northbridge MSRs in Dual Core Processors .....	332
12.10	Machine Check Architecture (MCA) .....	333
12.10.1	GART Table Walk Error Reporting .....	333
12.11	GART Register Restoration After S3 Resume .....	334
12.12	Register Settings in UMA Systems .....	334
12.13	Memory Map .....	335
12.13.1	I/O and Memory Type and Range Registers (IORRs, MTRRs) .....	335
12.13.2	Memory Map Registers (MMRs) .....	335
12.14	Access Routing And Type Determination .....	335
12.14.1	Memory Access to the Physical Address Space .....	335
12.14.2	Northbridge Processing of Accesses To Physical Address Space .....	337
12.15	Error Handling .....	337
12.15.1	Error Handling Mechanisms .....	337
12.15.2	Errors and Recommended Error Handling .....	338
12.16	Cache Initialization For Temporary Storage During Boot .....	340
12.17	Cache Testing and Programming .....	340
12.18	Memory System Configuration Registers .....	340
12.19	Processor ID .....	341
12.20	XSDT Table .....	341
12.21	Detect Target Operating Mode Callback .....	341
12.22	SMM Issues .....	342
<b>13</b>	<b>Processor Configuration Registers .....</b>	<b>343</b>
13.1	General Model-Specific Registers .....	343

13.1.1	System Software Registers .....	345
13.1.2	Memory Typing Registers .....	347
13.1.3	APIC Registers .....	364
13.1.4	Software Debug Registers .....	365
13.1.5	Performance Monitoring Registers .....	366
13.2	AMD Athlon™ 64 processor and AMD Opteron™ Processor Model-Specific Registers .....	366
13.2.1	Feature Registers .....	368
13.2.2	Identification Registers .....	375
13.2.3	Memory Typing Registers .....	375
13.2.4	I/O Range Registers .....	376
13.2.5	System Call Extension Registers .....	378
13.2.6	Segmentation Registers .....	380
13.2.7	Power Management Registers .....	382
13.2.8	IO and Configuration Space Trapping to SMI .....	385
13.2.9	Interrupt Pending Message Register .....	388
	<b>Glossary .....</b>	<b>391</b>
	<b>Index of Register Names .....</b>	<b>397</b>



## List of Figures

---

Figure 1.	Interleave Example (IntlvEn Relation to IntlvSel) .....	73
Figure 2.	8P Ladder Configuration .....	146
Figure 3.	8P Twisted Ladder Configuration.....	146
Figure 4.	Default SMM Memory Map .....	220
Figure 5.	A Four-Node Configuration.....	238
Figure 6.	High-Level P-state Transition Flow .....	272
Figure 7.	Example P-State Transition Timing Diagram .....	273
Figure 8.	Phase 1: Core Voltage Transition Flow .....	274
Figure 9.	Phase 2: Core Frequency Transition Flow.....	275
Figure 10.	Phase 3: Core Voltage Transition Flow .....	276



# List of Tables

Table 1.	Related Documents .....	28
Table 2.	Function 0 Configuration Registers .....	39
Table 3.	Function 1 Configuration Registers .....	66
Table 4.	Function 2 Configuration Registers .....	81
Table 5.	DRAM CS Base Address and DRAM CS Mask Registers.....	85
Table 6.	DRAM CS Base Address and DRAM CS Mask Registers Mismatched DIMM Support1 .....	85
Table 7.	Mapping Host Address Lines to Memory Address Lines with 64-Bit Interface (revision CG and earlier revisions).....	90
Table 8.	Mapping Host Address Lines to Memory Address Lines with 128-Bit Interface (revision CG and earlier revisions).....	90
Table 9.	Mapping Host Address Lines to Memory Address Lines with 64-Bit Interface (revision D and later revisions).....	91
Table 10.	Mapping Host Address Lines to Memory Address Lines with 128-Bit Interface (revision D and later revisions).....	92
Table 11.	Mapping Host Address Lines to Memory Address Lines with 64-Bit Interface Bank Swizzle Mode (Revision E and later revisions) .....	93
Table 12.	Mapping Host Address Lines to Memory Address Lines with 128-Bit Interface Bank Swizzle Mode (Revision E and later revisions) .....	93
Table 13.	Swapped physical address lines for interleaving with 64-bit interface (revision CG and earlier revisions).....	95
Table 14.	Swapped physical address lines for interleaving with 128-bit interface (revision CG and earlier revisions).....	96
Table 15.	Swapped physical address lines for interleaving with 64-bit interface (revision D and later revisions).....	96
Table 16.	Swapped physical address lines for interleaving with 128-bit interface (revision D and later revisions).....	96
Table 17.	Function 3 Configuration Registers .....	118
Table 18.	Error Code Field Formats.....	128
Table 19.	Transaction Type Bits (TT).....	128
Table 20.	Cache Level Bits (LL).....	128
Table 21.	Memory Transaction Type Bits (RRRR) .....	129
Table 22.	Participation Processor Bits (PP) .....	129
Table 23.	Time-Out Bit (T).....	129
Table 24.	Memory or I/O Bits (II).....	129
Table 25.	Northbridge Error Codes.....	130
Table 26.	Northbridge Error Status Bit Settings .....	132
Table 27.	ECC Syndromes.....	138
Table 28.	Chip Kill ECC Syndromes.....	139
Table 29.	Scrub Rate Control Values.....	141

Table 30.	XBAR Input Buffers .....	143
Table 31.	Default XBAR Command Buffer Allocation.....	143
Table 32.	Default Virtual Channel Command Buffer Allocation .....	144
Table 33.	An Example of a Non Default Virtual Channel Command Buffer Allocation .....	145
Table 34.	2P Dual Coherent HyperTransport Command Buffer Allocation.....	145
Table 35.	8P Outer Node Virtual Channel Command Buffer Allocation .....	146
Table 36.	8P Inner Node Virtual Channel Command Buffer Allocation.....	146
Table 37.	Recommended Flow Control Buffer Allocations in UMA systems .....	147
Table 38.	GART PTE Organization .....	157
Table 39.	Diode Offset Encodings .....	164
Table 40.	Trwt Values .....	173
Table 41.	RdPreamble Values .....	174
Table 42.	Unbuffered DIMM Support For 754-pin Lidded Micro PGA Package.....	175
Table 43.	Unbuffered SO-DIMM Support For 754-pin Lidless Micro PGA Package (Single Address/Control Bus Motherboard Implementation1).....	177
Table 44.	Unbuffered SO-DIMM Support For 754-pin Lidless Micro PGA Package (Dual Address/Control Bus Motherboard Implementation1) .....	178
Table 45.	Unbuffered DIMM Support For 939-pin Lidded Micro PGA Package.....	178
Table 46.	Unbuffered DIMM Support For Revision E 939-pin Lidded Micro PGA Package .....	179
Table 47.	SODIMM Support For Revision E 939-pin Lidded Micro PGA Package.....	182
Table 48.	Registered DIMM Support for 940-pin Lidded Micro PGA Package .....	183
Table 49.	DDR400 and Quad Rank Registered DIMM Support for 940-pin Lidded Micro PGA Package .....	184
Table 50.	Sources of Machine Check Errors.....	191
Table 51.	DC Error Codes.....	202
Table 52.	DC Error Status Bit Settings .....	203
Table 53.	Valid MC0_ADDR Bits.....	204
Table 54.	IC Error Codes .....	206
Table 55.	IC Error Status Bit Settings.....	206
Table 56.	Valid MC1_ADDR Bits.....	207
Table 57.	BU Error Codes.....	210
Table 58.	BU Error Status Bit Settings .....	211
Table 59.	Valid MC2_ADDR Bits.....	211
Table 60.	SMM Save State (Offset FE00–FFFFh) .....	221
Table 61.	SMM Entry State.....	224
Table 62.	SMM ASeg-Enabled Memory Types.....	231
Table 63.	SMM TSeg-Enabled Memory Types .....	232
Table 64.	APIC Register Summary .....	244
Table 65.	Valid Combinations of ICR Fields.....	254
Table 66.	Power Management Categories.....	261
Table 67.	ACPI-State Support by System Class .....	262
Table 68.	Required SMAF Code to Stop Grant Mapping.....	263
Table 69.	Low FID Frequency Table (< 1600 MHz).....	270
Table 70.	High FID Frequency Table (>= 1600 MHz).....	270



Table 71.	Sample VST Values .....	281
Table 72.	MVS Values .....	282
Table 73.	RVO Values .....	282
Table 74.	VID Code Voltages .....	283
Table 75.	IRT Values .....	283
Table 76.	_PSS Status Field .....	284
Table 77.	Performance State Block Structure .....	287
Table 78.	Configuration Register Settings for Power Management .....	289
Table 79.	FADT Table Entries .....	292
Table 80.	Northbridge MSRs .....	333
Table 81.	General MSRs .....	343
Table 82.	AMD Athlon™ 64 Processor and AMD Opteron™ Processor MSRs .....	367
Table 83.	FID Code Translations .....	383



## Revision History

---

Date	Rev.	Description
October 2005	3.28	<p>Updated Table 5.</p> <p>Updated 3.5.7 "Address to Node/Chip Select Translation".</p> <p>Updated 3.5.8.2 "Memory Hoisting For Revision E and Later Revisions".</p> <p>Updated 3.5.11.2 "DRAM Configuration High Register".</p> <p>Updated 3.6.8 "XBAR Flow Control Buffers".</p> <p>Updated Table 37.</p> <p>Updated 3.6.18 "HyperTransport™ FIFO Read Pointer Optimization Register".</p> <p>Updated 3.6.19 "Thermtrip Status Register".</p> <p>Added Table 46.</p> <p>Added Table 47.</p> <p>Updated Table 49 with quad rank RDIMM support.</p> <p>Updated Table 57.</p> <p>Updated Table 78.</p> <p>Updated Chapter 11, "CPUID".</p> <p>Added 12.7 "APIC ID Assignment Requirements".</p>

Date	Rev.	Description
April 2005	3.26	<p>Added Revision E functionality.</p> <p>Corrected 3.5.7 “Address to Node/Chip Select Translation”.</p> <p>Corrected 3.5.8.2 “Memory Hoisting For Revision E and Later Revisions”.</p> <p>Updated 3.6.8 “XBAR Flow Control Buffers” with recommendation for 8P buffer allocations.</p> <p>Clarified DiodeOffsetSignBit description in 3.6.19 “Thermtrip Status Register”</p> <p>Modified 4.1.2.5 “Maximum Asynchronous Latency”.</p> <p>Updated Chapter 7, “HyperTransport™ Technology Configuration and Enumeration”.</p> <p>Updated 9.8.1 “Chipset Configuration for Power Management”.</p> <p>Updated Table 78.</p> <p>Added 10.2.1 “Performance Monitor Events”.</p> <p>Added Chapter 11, “CUID”.</p> <p>Added 12.16 “Cache Initialization For Temporary Storage During Boot”</p> <p>Modified 12.6 “Setup for Dual Core Processors”.</p> <p>Modified 12.12 “Register Settings in UMA Systems”.</p> <p>Added 12.15.2.5 “Watchdog Timeout Errors”.</p> <p>Clarified 13.2.9 “Interrupt Pending Message Register”.</p> <p>Modified 13.2.1.4 “NB_CFG Register”.</p> <p>Modified 13.2.1.3 “HWCR Register”.</p>

Date	Rev.	Description
January 2005	3.24	<p>Modified Table 78.</p> <p>Modified 3.6.5.2 "Chip Kill".</p> <p>Clarified LinkFail (Function 0, Offsets 84h, A4h, C4h).</p> <p>Clarified Cpu1En (Function 0, Offsets 68h).</p> <p>Clarified SYSCFG (MSR C001_0010h).</p> <p>Added FFXSR, LMSLE (MSR C000_0080h).</p> <p>Modified CS[7/6,5/4,3/2,1/0] (Function 2, Offset 80h) definition.</p> <p>Added DRAM mapping information (Table 9, Table 10, Table 15, and Table 16).</p> <p>Added PwrDownCtl, UpperCSMap, DualDimmEn (Function 2, Offset 90h).</p> <p>Added DllSpeedOverride, DllSpeed (Function 2, Offset 98h).</p> <p>Added MCA Error Information to: Table 51, Table 52, Table 54, Table 55, Table 57, Table 58.</p> <p>Corrected "NMI Mask" location in "SMM State Save Definition".</p> <p>Modified "LDTi Frequency/Revision Registers".</p> <p>Modified 4.1.2.3 "Trwt".</p> <p>Modified "Thermtrip Status Register".</p> <p>Added 9.5.2 "BIOS Requirements for P-State Transitions in Systems Using Dual Core Processors".</p> <p>Added 9.5.6.3 "Dual Core P-state Transition Requirements".</p> <p>Modified 9.6 "ACPI 2.0 Processor P-State Objects".</p> <p>Modified 9.7 "BIOS Support for AMD PowerNow!™ Software with Legacy Operating Systems".</p> <p>Modified 10.2 "Performance Event-Select Registers".</p> <p>Modified 12.6 "Setup for Dual Core Processors".</p> <p>Added 12.9.1 "Software Considerations for Accessing Northbridge MSRs in Dual Core Processors".</p> <p>Modified 13.2.9 "Interrupt Pending Message Register".</p>

<b>Date</b>	<b>Rev.</b>	<b>Description</b>
November 2004	3.22	<p>Added MP P-state requirements to Chapter 9, "Power and Thermal Management".</p> <p>Modified "DRAM Configuration Registers".</p> <p>Modified "Free List Buffer Count Register".</p> <p>Modified "DRAM Delay Line Register".</p> <p>Modified "Power Management Mode (PMM) Value".</p> <p>Modified "MCA NB Configuration Register".</p> <p>Changed bit 27 to reserved SBZ in "Clock Power/Timing High Register".</p> <p>Modified "DRAM Hole Address Register".</p> <p>Modified Table 78.</p> <p>Added Table 6.</p> <p>Modified "Node ID Register".</p> <p>Modified 9.5.1 "BIOS Requirements for P-State Transitions".</p> <p>Modified "LDTi Frequency/Revision Registers".</p> <p>Modified 3.4.4 "DRAM Address Map".</p> <p>Modified Table 45.</p> <p>Modified 9.2.2 "C2 and C3".</p> <p>Modified 4.1.2.5 "Maximum Asynchronous Latency".</p> <p>Modified 4.1.2.6 "Read Preamble Time".</p> <p>Modified 12.8 "Processors With Different Revisions or Maximum Frequencies in Multiprocessor Systems".</p>
September 2004	3.20	<p>Updated "Multiprocessing Capability Detection".</p> <p>Updated "Setup for Dual Core Processors".</p> <p>Updated "One-Node Coherent HyperTransport™ Technology Initialization".</p> <p>Updated Table 78.</p> <p>Updated "DRAM Configuration Registers".</p>

Date	Rev.	Description
August 2004	3.18	<p>Modified "Maximum DRAM Speed as a Function of Loading".</p> <p>Modified "IOTRAP_ADDRi Registers".</p> <p>Added "Address to Node/Chip Select Translation".</p> <p>Modified "Memory Hoisting".</p> <p>Added "Memory Hoisting For Revision E and Later Revisions".</p> <p>Added "C3 and AltVID".</p> <p>Modified Table 78.</p> <p>Modified "DIMM Matching Algorithm".</p> <p>Added "DRAM Hole Address Register".</p> <p>Modified "Power Management Control Registers".</p> <p>Modified "LDTi Link Control Registers".</p> <p>Modified "Clock Power/Timing Low Register".</p> <p>Modified "Clock Power/Timing High Register".</p> <p>Modified "Northbridge Capabilities Register".</p> <p>Modified "MCA NB Configuration Register".</p> <p>Modified "NB_CFG Register".</p> <p>Modified "DRAM Configuration Low Register".</p> <p>Modified "DRAM Configuration High Register".</p> <p>Modified "DRAM Bank Address Mapping Register".</p> <p>Modified "DRAM Delay Line Register".</p> <p>Modified "FIDVID_CTL Register".</p> <p>Modified "FIDVID_STATUS Register".</p> <p>Added "Setup for Dual Core Processors".</p>
June 2004	3.16	<p>Clarified RMW support in "MCA NB Configuration Register".</p> <p>Modified Table 48.</p> <p>Clarified LDTSTOP_L Minimum assertion time in "Chipset Configuration for Power Management" and "Initializing Link Width and Frequency".</p> <p>Clarified that the backbone first rule is used to generate the broadcast routing table in "Broadcast Routing Table".</p> <p>Clarified P-state requirements for mixed steppings in "Processors With Different Revisions or Maximum Frequencies in Multiprocessor Systems".</p> <p>Clarified L2 Cache requirements for mixed steppings in "Processors With Different Revisions or Maximum Frequencies in Multiprocessor Systems".</p> <p>Modified "DRAM Configuration Registers"</p> <p>Added "HyperTransport™ Link Detection"</p> <p>Added "Error Handling"</p>

Date	Rev.	Description
April 2004	3.14	<p>Added "GART Register Restoration After S3 Resume".</p> <p>Added "Register Settings in UMA Systems".</p> <p>Added CPU_CLK_UNHALTED performance monitor.</p> <p>Modified reset value for "MCi_STATUS—Machine Check Status Registers" and "MC0_STATUS—DC Machine Check Status Register".</p> <p>Added ECC syndromes for ECC check bits to "Normal ECC".</p> <p>Added requirement for P-state changes to the algorithm in "Processors With Different Revisions or Maximum Frequencies in Multiprocessor Systems".</p> <p>Added requirement for DP and MP systems with different model numbers or different operating frequencies to "Multiprocessing Capability Detection".</p> <p>Corrected SMM_MASK address (MSR C001_0113h) in "Access Routing And Type Determination".</p> <p>Added reference to "HyperTransport™ Link Frequency Selection" to "Initializing Link Width and Frequency".</p>
March 2004	3.12	<p>Added "Processors With Different Revisions or Maximum Frequencies in Multiprocessor Systems".</p> <p>Modified "HyperTransport™ Link Frequency Selection" and "Multiprocessing Capability Detection".</p> <p>Corrected Function 3, Offset 80h register value in Table 78.</p> <p>Corrected MC4_CTL address in "GART Table Walk Error Reporting".</p> <p>Clarified "Interrupt Pending Message Register".</p>
January 2004	3.08	<p>Added "GART Table Walk Error Reporting".</p> <p>Added "Access Routing And Type Determination".</p> <p>Added "Processor ID".</p> <p>Added requirement for extended configuration space memory type in "Extended Configuration Space Access".</p> <p>Added "Memory Hoisting".</p> <p>Clarified D_DRV (Function 2, Offset 90h).</p> <p>Added Scratch Register (Function 2, Offset 9Ch).</p> <p>Added recommendation for flow control buffer settings in a UMA system to "XBAR Flow Control Buffers".</p> <p>Clarified ThermtpSense (Function 3, Offset E4h).</p> <p>Added requirement for 3-DIMM unbuffered configurations to "Processor Performance States".</p> <p>Clarified MCi_STATUS_WREN (MSR C001_0015h).</p> <p>Added "DIMM Matching Algorithm".</p> <p>Added Enable 2T timing (En2T) bit to "DRAM Configuration Low Register".</p> <p>Added "Interrupt Pending Message Register".</p>



Date	Rev.	Description
September 2003	3.06	<p>Clarified 32ByteEn (Function 2, Offset 90h) programming in “DRAM Configuration” section.</p> <p>Removed SleepVIDCnt (Function 3, Offset D4h) and added a programming requirement.</p> <p>Added reference to the <i>AMD Athlon™ 64 Data Sheet</i>, order# 24659 for 3 DIMM configuration to “Programming Interface” section.</p> <p>Corrected requirement for supported DIMMs in “Registered or Unbuffered DIMMs” section.</p> <p>Corrected RdPreamble values for 3 DIMM configuration in “Read Preamble Time” section.</p> <p>Added RdPreamble and Trwt values for registered DDR400.</p> <p>Added EnRefUseFreeBuf and DisCohLdtCfg to NB_CFG register (MSR C001_001Fh).</p> <p>Cleanup related to AMD Athlon™ 64 and AMD Opteron™ use.</p>
July 2003	3.04	<p>Added examples to “DRAM Address Mapping in Interleaving Mode” and “DRAM Address Mapping in Non-Interleaving Mode” sections.</p> <p>Added recommendation for setting “HyperTransport™ FIFO Read Pointer Optimization Register”.</p> <p>Added “Spurious Interrupts Caused by Timer Tick Interrupt” section.</p> <p>Added “XSDT Table” section.</p> <p>Added “Detect Target Operating Mode Callback” section.</p> <p>Modified “Multiprocessing Capability Detection” section.</p> <p>Added requirements for AMD Cool'n'Quiet™ technology and BIOS P-state transitions to “Power and Thermal Management” chapter.</p> <p>Added ClkRampHyst (Function 3, Offset D4h) setting to “Power and Thermal Management” chapter.</p> <p>General cleanup in “Power and Thermal Management” chapter.</p> <p>Added “Twtr (Write to Read Delay)” section.</p> <p>Added a requirement to “GART Aperture Base Register”.</p> <p>Added a requirement for GartEn (Function 3, Offset 90h) programming.</p> <p>Removed recommendation for extended range temperature sensors in DiodeOffset (Function 3, Offset E4h) definition.</p> <p>Added a requirement for extended configuration space access to “Memory-Mapped I/O Address Map Registers” section.</p>

<b>Date</b>	<b>Rev.</b>	<b>Description</b>
May 2003	3.02	Added "ECC and Chip Kill Error Checking and Correction" section. Clarified "Scrub Control Register" and "DRAM Scrub Address Registers" sections. Clarified LinkFail (Function 0, Offsets 84h, A4h, C4h). Clarified node interleaving in "DRAM Address Map" section. Clarified and corrected DRAM address interleaving in "DRAM CS Base Address Registers" and "DRAM Bank Address Mapping Register" sections. Clarified "tCL (CAS Latency)" section. Added "piggyback scrubbing" description to "Sources of Machine Check Errors" section. Clarified Table 50, "Sources of Machine Check Errors," on page 191. Clarified MCA NB PCI configuration register to MCA MSR mapping in "Northbridge (NB)" section. Corrected MaxLVTEEntry (APIC_VER Register, Offset 30h) default value. General cleanup in "Power and Thermal Management" chapter. Added "HyperTransport™ Link Frequency Selection" section.
April 2003	3.00	Initial Public release.

# 1 Introduction and Overview

---

The *BIOS and Kernel Developer's Guide for the AMD Athlon™ 64 and AMD Opteron™ Processors* is intended for programmers involved in the development of low-level BIOS (basic input/output system) functions, drivers, and operating system kernel modules. It assumes previous experience in microprocessor programming, as well as fundamental knowledge of legacy x86 and AMD64 microprocessor architecture. The reader should also have previous experience in BIOS or OS kernel design issues, as related to microprocessor systems, and a familiarity with various platform technologies, such as DDR, HyperTransport™ technology, and JTAG.

## 1.1 About This Guide

This guide covers the *implementation-specific* features of AMD Athlon™ 64 and AMD Opteron™ Processors, as opposed to architectural features. AMD64 architectural features include the AMD64 technology, general-purpose, multimedia, and x87 floating-point registers, as well as other programmer-visible features defined to be constant across all processors. A subset of implementation-specific features are not defined by the processor architectural specifications. These implementation-specific features may differ in various details from one implementation to the next.

**Note:** *The term processor in this document refers to AMD Athlon™ 64 processor architecture and AMD Opteron™ processor architecture. This document covers both classes of devices. For details about differences between them, see the AMD Athlon™ 64 Processor Data Sheet, order# 24659 and the AMD Opteron™ Processor Data Sheet, order# 23932.*

The implementation-specific features covered in the following chapters include:

- Model-specific registers
- Processor initialization
- Integrated memory system configuration
- HyperTransport technology fabric initialization
- Performance monitoring and special debug features
- DRAM configuration
- Machine check error codes
- Thermal and power management

## 1.2 Related Documents

The references listed in Table 1 may prove invaluable towards a complete understanding of the subject matter in this volume.

**Table 1. Related Documents**

Title	Order#
<i>AMD64 Architecture Programmer's Manual, Volume 1, Application Programming</i>	24592
<i>AMD64 Architecture Programmer's Manual, Volume 2, System Programming</i>	24593
<i>AMD64 Architecture Programmer's Manual, Volume 3, General-Purpose and System Instructions</i> <i>AMD64 Architecture Programmer's Manual, Volume 4, 128-Bit Media Instructions</i> <i>AMD64 Architecture Programmer's Manual, Volume 5, 64-Bit Media and x87 Floating-Point Instructions</i>	24594 (three-volume kit)
<i>AMD Opteron™ Processor Data Sheet</i>	23932
<i>AMD Athlon™ 64 Processor Data Sheet</i>	24659
<i>AMD Processor Recognition Application Note</i>	20734
<i>CPUID Guide for AMD Athlon™ 64 and AMD Opteron™ Processors</i>	25481
<i>Revision Guide for the AMD Athlon™ 64 and AMD Opteron™ Processors</i>	25759
<i>HyperTransport™ I/O Link Specification, Rev. 1.03</i>	<a href="http://www.hypertransport.org">http://www.hypertransport.org</a>

## 1.3 Conventions and Definitions

Some of the following definitions assume a knowledge of the legacy x86 architecture. See Table 1 for documents that include information on the legacy x86 architecture.

Additional definitions are provided in the glossary at the end of this book, beginning on page 391. That glossary includes the most important terminology of the AMD64 architecture.

### 1.3.1 Notation

**1011b.** A binary value. In this example, a 4-bit value is shown.

**F0EAh.** A hexadecimal value. In this example, a 2-byte value is shown.

**[1,2].** A range that includes the left-most value (in this case, 1) but excludes the right-most value (in this case, 2).

**7–4.** A bit range, from bit 7 to 4, inclusive. The high-order bit is shown first.

**#GP(0).** Notation indicating a general-protection exception (#GP) with error code of 0.

**CR0–CR4.** A register range, from register CR0 through CR4, inclusive, with the low-order register first.

**CR0.PE = 1.** Notation indicating that the PE bit of the CR0 register has a value of 1.

**DS:rSI.** The contents of a memory location whose segment is DS and whose byte address is located in the rSI register.

**EFER.LME = 0.** Notation indicating that the LME bit of the EFER register has a value of 0.

**FF /0.** Notation indicating that FF is the first byte of an opcode and a sub-opcode field in the MODRM byte has a value of 0.

**DramEn.** Notation indicating that the bit is an enable ("En" or "EN" is part of the name) and that a 1 specifies that the function is enabled.

**MemClrDis.** Notation indicating that the bit is a disable ("Dis" or "DIS" is part of the name) and that a 1 specifies that the function is disabled.

## 1.4 Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors

Some changes to the register set are introduced with different silicon revisions. Refer to the *Revision Guide for AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25759 for information about how to identify different processor revisions. The following summarizes register changes after the initial revision.

### Revision C.

- MSR C001\_0043h (ThermTrip\_STATUS Register) deleted.
- Function 2, Offset 90h, DramEn (bit 10), MemClrStatus (bit 11) added.
- Function 2, Offset CCh, DisTscCapture (bit 30) added.
- Function 3, Offset 70h, DispRefReq (bits 21-20) added.
- Function 3, Offset 74h, DispRefReq (bits 22-20) added.
- MSR C001\_001Fh (NB\_CFG register), DisDatMsk (bit 36), EnRefUseFreeBuf (bit 9) added.
- Function 3, Offset 90h, DisGartTblWlkPrb (bit 6) added.
- Function 3, Offset D4h, ClkRampHyst (bits 10-8) added.
- MSR C001\_0015h (HWCR register), HLTXSycen (bit 12) added.
- MSR C001\_0010h (SYSCFG register), ClVicBlkEn (bit 11) deleted.
- MSRs C001\_0050h, C001\_0051h, C001\_0052h, C001\_0053h (IOTRAP\_ADDR*i* registers), and MSR C001\_0054h (IOTRAP\_CTL register) added.

**Revision CG.**

- Function 2, Offset 90h, En2T (bit 28) added.
- MSR C001\_0055h (Interrupt Pending Message register) added.

**Revision D.**

- Function 2, Offset 90h, PwrDownCtl (bits 31-30), UpperCSMap (bit 29), DualDimmEn (bit 9) added.
- Function 2, Offset 98h, DllSpeedOverride (bit 30), DllSpeed (bit 29) added.
- MSR C000\_0080h (EFER register), LMSLE (bit 13), FFXSR (bit 14) added.
- Function 3, Offset E4h, Tcasemax (bits 28-25) and DiodeOffsetSignBit (bit 24) added.
- Function 2, Offset C0h, FlushTcbDbReq (bit 15) added. CPUID Fn[8000\_0001][EDX] FFXSR (bit 25) added.

**Revision E**

- Function 0, Offset 84h, A4h, C4h LdtStopRcvDis (bit 11), LdtStopTriClkOvr (bit 11) added.
- Function 3, Offset D4h, ClkRampHyst (bit 11), ReConDel (bits 15-12) added.
- Function 1, Offset F0h (Dram Hole Address Register) added.
- Function 3, Offset E8h, CmpCap (bits 13-12) added.
- Function 3, Offset 44h, NbMcaToMstCpuEn (bit 27) added.
- MSR C001\_001Fh (NB Configuration register), DisThmIPfMonSmiIntr (bit 43), DisUsSysMgtRqToNLdt (bit 45), InitApicIdLoCpuIdLo (bit 54) added.
- Function 3, Offset D8h, AltVid (bits 24-20) added.
- Function 3, Offset 80h/84h, AltVidEn (bit 3) added.
- MSR C001\_0042h (FIDVID\_STATUS register), MinVID (bits 61-56) added.
- Function 2, Offset 94h, OddDivisorCorrect (bit 31), MemDQDrvStren (bits 14-13) added.
- Function 2, Offset 90h, Burst2Opt (bit 5), Mod64BitMux (bit 6), PwrDwnTriEn (bit 7) added.
- Function 2, Offset 90h, PwrDownCtl (bits 31-30) modified.
- Function 2, Offset 98h, AltVidC3MemClkTriEn (bit 27), CompPwrSaveEn (bit 28) added.
- Function 2, Offset 80h, BankSwizzleMode (bit 30) added.
- APIC 340h (PERF\_CNT\_LVT) MsgType (bits 10-8) modified.
- CPUID Fn[0000\_0001][EBX] Logical Processor Count (bits 23-16) added.

- CPUID Fn[0000\_0001][ECX] SSE3 (bit 0) added.
- CPUID Fn[8000\_0001][ECX] CMPLegacy (bit 1) added.
- CPUID Fn[0000\_0001][EDX] HTT (bit 28) added.

•





## 2 Processor Initialization and Configuration

---

Each AMD Athlon™ 64 and AMD Opteron™ Processor based system has one processor with its HyperTransport™ link connected to a HyperTransport™ I/O hub. When a reset signal is applied, this processor is initialized as the bootstrap processor (BSP). In a multiple processor system, any other processors are initialized as application processors (APs). The BSP begins executing code from the reset vector (0xFFFFFFFF), while each of the APs waits for its Request Disable bit to clear to 0. An AP does not fetch code until its Request Disable bit is cleared. Both BSP and AP operate in 16-bit Real mode after reset. The BSP has the Boot Strap Processor bit set in its APIC\_BASE (001Bh) MSR, and each AP has this bit cleared.

The processor node is addressed by its Node ID on the HyperTransport link and can be accessed with a device number in the PCI configuration space on Bus 0. The Node ID 0 is mapped to Device 24, the Node ID 1 is mapped to Device 25, and so on. The BSP is initialized with Node ID 0 after reset, and all APs are initialized with Node ID 7.

The BSP is ready to access its Northbridge and memory controller after its routing table is enabled. The APs are not accessible from the BSP until the links and the routing table are configured.

The initial processor states are described in the “Processor Initialization State” section of the *AMD64 Architecture Programmer's Manual: Volume 2, System Programming*.

### 2.1 Bootstrap Processor Initialization

The BSP is responsible for the execution of the BIOS Power-On Self Test (POST) and initialization of APs. The BSP must perform the following tasks:

- AP detection and routing table initialization
- Noncoherent HyperTransport device initialization
- Link width and frequency initialization
- Memory controller initialization on all processor nodes
- Address map table initialization

#### 2.1.1 Detecting AP and Initializing Routing Table

The AMD Opteron™ processor has three HyperTransport links; therefore, an AMD Opteron™ system can have from one to eight processors. The BSP must detect all APs in the system, starting from its adjacent processor. Since the APs are initialized at Node 7, the BSP must set entry 7 of its Routing Table before the AP can be accessed. The adjacent AP will respond to a PCI configuration

read if it is present. A new Node ID can be assigned to the AP after the routing information is updated in the routing table. Chapter 7, “HyperTransport™ Technology Configuration and Enumeration” explains the steps for enumerating and initializing routing tables in one-processor (UP), two-processor (DP), and  $n$  processor systems.

The AMD Athlon™ 64 processor has one HyperTransport link. Because one link must be connected to the HyperTransport I/O hub, an AMD Athlon™ 64 system can only be used in a uniprocessor system.

## 2.1.2 Initializing Noncoherent HyperTransport™ Technology Devices

Once all APs have been detected and the routing tables and link control registers have been initialized, the BSP must initialize noncoherent HyperTransport technology devices.

A noncoherent HyperTransport technology device has either one or two links. A tunnel device has two links and a terminal device has one link. The HyperTransport I/O hub is a typical example of a terminal noncoherent HyperTransport device. The noncoherent HyperTransport device is identified by its Unit ID in a noncoherent HyperTransport link and can be accessed with a device number in the PCI configuration space. The device number in PCI space is the same as the Unit ID assigned to the noncoherent HyperTransport device.

After reset, the Unit ID of each noncoherent HyperTransport technology device is initialized with a value of 0. When a PCI configuration read is performed from the BSP through a noncoherent HyperTransport link, the noncoherent HyperTransport device connected to the port responds. A new non-zero Unit ID value must be assigned to this device. The BIOS continues this process until no device responds at Device 0. The bus number range must be set to the noncoherent HyperTransport link on the BSP and in the address map table prior to the detection. The link control and status registers of a noncoherent device are implemented in the capability register block.

The noncoherent HyperTransport technology devices connected to a port on the AP node can be detected in the same way. Again, the bus number range must be set to the noncoherent HyperTransport link on this AP and in the address map table. The noncoherent device can be detected with the starting bus number, Device 0 on the PCI configuration space.

A noncoherent HyperTransport technology device may use more than one Unit ID. The new ID assigned to a device is its starting ID, the Base Unit ID (BaseUID). The next logic device in this noncoherent HyperTransport device can be identified with BaseUID + 1, and so on. The Unit ID Count field in the Capabilities register indicates how many Unit IDs this device uses.

## 2.1.3 Initializing Link Width and Frequency

The HyperTransport link width and frequency are initialized between the adjacent coherent and/or noncoherent HyperTransport technology devices during the reset sequence. After AP and noncoherent HyperTransport device detection, the link width and frequency can be changed based on the capability of the adjacent devices or the implementation of the system. See “HyperTransport™ Link Frequency Selection” on page 328 for more information. To make the new link width and

frequency take effect, an LDTSTOP\_L needs to be asserted or a warm reset must be performed. The BIOS must guarantee that LDTSTOP\_L assertion for link width and frequency changes does not occur within 200  $\mu$ s of reset. LDTSTOP\_L must be asserted for a minimum of 2  $\mu$ s for link width and frequency changes.

## **2.1.4 Initializing the Memory Controller on All Processor Nodes**

The BSP is responsible for configuring the memory controller on all processor nodes and for initializing the DRAM map table. Chapter 4, “DRAM Configuration,” describes the functionality of the memory controller and the steps required to initialize memory.

After memory configuration, the memory address MSRs must be set accordingly. The Top of Memory MSR is used for the top of DRAM below 4 Gbytes, and TOP\_MEM2 is set to the top of DRAM above 4 Gbytes.

## **2.1.5 Initializing the Address Map Table**

Each processor node has an address map table. This table contains the address map for the DRAM area, the address map for PCI memory spaces (MMIO), the address map for PCI I/O spaces, and the bus number range for each noncoherent HyperTransport link. The address map table must be duplicated on all processors in a system.

## **2.2 Application Processor Initialization**

The application processor (AP) waits until its request disable bit is cleared to 0. The BIOS may clear this bit for an AP as soon as the AP node detection is completed and the routing tables are initialized, or just before the AP register initialization.

When the request disable bit is cleared, the corresponding AP starts to fetch code from the reset vector (0xFFFFFFFF). The BSP bit in AP APIC\_BASE register is cleared at reset, so it can be used to terminate the AP execution after the initial APIC initialization.

The AP register initialization is the same as that of other AMD x86 processor families, such as the AMD Athlon™ processors.

## **2.3 BIOS Requirement for 64-Bit Operation**

In general, the BIOS need not be aware of 64-bit mode in POST. There are no additional requirements to support 64-bit mode. The operating system determines whether to enable 64-bit mode after the BIOS invokes the operating system loader in legacy mode. There are two areas that need additional explanation:

- Sizing and testing memory above 4 Gbytes
- Using BIOS callback when in 64-bit mode

### **2.3.1 Sizing and Testing Memory above 4 Gbytes**

AMD Athlon™ 64 and AMD Opteron™ Processors have extended physical address extension (PAE) to support a 40-bit address space. Thus, the BIOS can set up a 32-bit page table that allows it to size and test all physical memory in the system.

### **2.3.2 Using BIOS Callbacks in 64-Bit Mode**

A BIOS callback is a mechanism that allows an operating system to call a service routine in the BIOS. BIOS callbacks on an AMD Athlon™ 64 processor or an AMD Opteron™ processor platform in 64-bit mode are not supported by AMD64 operating systems. An operating system loader must be invoked in legacy mode with paging disabled, so that the loader can use 16/32-bit BIOS callbacks.

ACPI code and operating system drivers are not affected by AMD64 operating systems that do not use a BIOS callback in 64-bit mode. ACPI is coded in ASL, which is not affected by the operating system mode. AMD64 operating system drivers are not allowed to use ROM callbacks.

## 3 Memory System Configuration

### 3.1 Configuration Space Accesses

The AMD Athlon™ 64 and AMD Opteron™ Processors implement configuration space as defined in the PCI Local Bus Specification, Rev. 2.2, and the HyperTransport™ I/O Link Specification, Rev. 1.03.

Both coherent HyperTransport links and the compatibility bus (the bus to the HyperTransport I/O hub) are accessed through Bus 0. Configuration accesses to Bus 0, devices 0 to 23, are transmitted to the compatibility noncoherent HyperTransport bus. Coherent HyperTransport device configuration space is accessed by using Bus 0, devices 24 to 31, where Device 24 corresponds to Node 0 and Device 31 corresponds to Node 7.

All configuration cycles are type 1 on coherent HyperTransport links. When transmitted down a noncoherent HyperTransport chain by the host bridge, configuration cycles are translated to type 0 if they target the noncoherent HyperTransport bus immediately behind the host bridge. If they target a subordinate bus, they remain as type 1.

Accesses to memory system configuration registers are controlled through the Configuration Address register (see “Configuration Address Register” on page 37) and the Configuration Data register (see “Configuration Data Register” on page 38), which can be accessed through I/O reads/writes to addresses 0CF8h and 0CFCh, respectively.

#### 3.1.1 Configuration Address Register

Writes to the Configuration Address register specify the target of a configuration access in terms of the bus, device, function, and register. Access to the Configuration Address register must be full doubleword reads or writes.

To access one of the memory system configuration registers defined in this chapter, the bus number should be 0, the device number should be the target processor node number plus 24, and the function and register number should be set based on the register function and offset as specified in this chapter.

When the Enable bit of the Configuration Address register is set, reads and writes to the Configuration Data register access the register specified in the Configuration Address register.

#### Configuration Address Register

0CF8h (doubleword)

31	24	23	16	15	11	10	8	7	2	1	0
EnReg	reserved			BusNum		DevNum		FuncNum	RegNum		reserved

Bit	Mnemonic	Function	R/W
31	EnReg	Enable Register	R/W
30–24	reserved		R
23–16	BusNum	Bus Number	R/W
15–11	DevNum	Device Number	R/W
10–8	FuncNum	Function Number	R/W
7–2	RegNum	Register Number	R/W
1–0	reserved		R

## Field Descriptions

**Register Number (RegNum)**—Bits 7–2. Specifies the doubleword offset of the configuration address.

**Function Number (FuncNum)**—Bits 10–8. Specifies the function number of the configuration address.

**Device Number (DevNum)**—Bits 15–11. Specifies the device number of the configuration address.

**Bus Number (BusNum)**—Bits 23–16. Specifies the bus number of the configuration address.

**Enable (EnReg)**—Bit 31. When this bit is set, aligned doubleword accesses to the Configuration Data Register result in configuration-space transactions.

0 = Configuration transactions disabled.

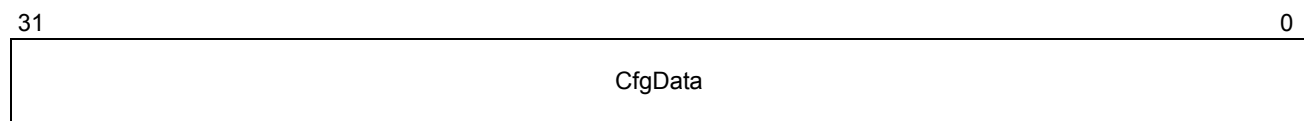
1 = Configuration transactions enabled.

### 3.1.2 Configuration Data Register

Accesses to the Configuration Data Register are translated to configuration-space transactions at the address specified by the Configuration Address Register.

#### Configuration Data Register

0CFCh (Doubleword)



Bits	Mnemonic	Function	R/W
31–0	CfgData	Configuration Data	R/W

## Field Descriptions

**Configuration Data (CfgData)**—Bits 31–0.

## 3.2 Memory System Configuration Registers

The AMD Athlon™ 64 and AMD Opteron™ Processors implement memory system configuration registers in the “Northbridge” block of the processor. These registers are mapped into the coherent HyperTransport technology configuration space (Bus 0, device numbers 24–31). The number of devices implemented is equal to the number of processor nodes in the system. Configuration space Device 24 corresponds to Node 0 and is normally the bootstrap processor (BSP). Configuration space device numbers 25 through 31 correspond to nodes 1 through 7, which may or may not be implemented.

The processor implements configuration registers in PCI configuration space using the following four headers:

- Function 0: HyperTransport technology configuration (see page 39)
- Function 1: Address map configuration (see page 66)
- Function 2: DRAM configuration (see page 80)
- Function 3: Miscellaneous configuration (see page 117)

Each of these functions are detailed in the following sections.

**Note:** Contents of some fields in the headers and HyperTransport technology capability blocks are maintained through a warm reset. If not specified otherwise, a field is initialized by a warm reset.

## 3.3 Function 0—HyperTransport™ Technology Configuration

**Table 2. Function 0 Configuration Registers**

Offset	Register Name				Reset	Access	Description
00h	Device ID		Vendor ID (AMD)		1100_1022h	RO	page 41
04h	Status		Command <sup>1</sup>		0010_0000h	RO	page 42
08h	Base Class Code	Subclass Code	Program Interface	Revision ID	0600_0000h	RO	page 42
0Ch	BIST	Header Type	Latency Timer	Cache Line Size	0080_0000h	RO	page 43
<b>Notes:</b> <ol style="list-style-type: none"> <li>1. The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.</li> <li>2. Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.</li> </ol>							

**Table 2. Function 0 Configuration Registers (Continued)**

Offset	Register Name				Reset	Access	Description
10h	Base Address 0				0000_0000h	RO	
14h	Base Address 1				0000_0000h	RO	
18h	Base Address 2				0000_0000h	RO	
1Ch	Base Address 3				0000_0000h	RO	
20h	Base Address 4				0000_0000h	RO	
24h	Base Address 5				0000_0000h	RO	
28h	Card Bus CIS Pointer				0000_0000h	RO	
2Ch	Sub-System ID		Sub-System Vendor ID		0000_0000h	RO	
30h	ROM Base Address				0000_0000h	RO	
34h	Capabilities Pointer				page 43	RO	page 43
38h	reserved				0000_0000h	RO	
3Ch	Max Latency	Min GNT	Int Pin	Int Line	0000_0000h	RO	
40h	Routing Table Node 0				0001_0101h	RW	page 44
44h	Routing Table Node 1				0001_0101h	RW	page 44
48h	Routing Table Node 2				0001_0101h	RW	page 44
4Ch	Routing Table Node 3				0001_0101h	RW	page 44
50h	Routing Table Node 4				0001_0101h	RW	page 44
54h	Routing Table Node 5				0001_0101h	RW	page 44
58h	Routing Table Node 6				0001_0101h	RW	page 44
5Ch	Routing Table Node 7				0001_0101h	RW	page 44
60h	Node ID				0000_0000h	RW	page 46
64h	Unit ID				0000_00E4h	RW	page 47
68h	HyperTransport™ Transaction Control				0F00_0000h	RW	page 48
6Ch	HyperTransport™ Initialization Control				page 53	RW	page 53
80h	LDT0 Capabilities				page 54	RO	page 54
84h	LDT0 Link Control				0011_0000h	RW	page 56
88h	LDT0 Frequency/Revision				page 60	RW	page 60
8Ch	LDT0 Feature Capability				0000_0002h	RO	page 61
90h	LDT0 Buffer Count				page 62	RW	page 62
94h	LDT0 Bus Number				0000_0000h	RW	page 64
98h	LDT0 Type				page 65	RO	page 65
A0h	LDT1 Capabilities				page 54	RO	page 54

**Notes:**

1. The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.
2. Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.



**Table 2. Function 0 Configuration Registers (Continued)**

Offset	Register Name	Reset	Access	Description
A4h	LDT1 Link Control	0011_0000h	RW	page 56
A8h	LDT1 Frequency/Revision	page 60	RW	page 60
ACh	LDT1 Feature Capability	0000_0002h	RO	page 61
B0h	LDT1 Buffer Count	page 62	RW	page 62
B4h	LDT1 Bus Number	0000_0000h	RW	page 64
B8h	LDT1 Type	page 65	RO	page 65
C0h	LDT2 Capabilities	page 54	RO	page 54
C4h	LDT2 Link Control	0011_0000h	RW	page 56
C8h	LDT2 Frequency/Revision	page 60	RW	page 60
CCh	LDT2 Feature Capability	0000_0002h	RO	page 61
D0h	LDT2 Buffer Count	page 62	RW	page 62
D4h	LDT2 Bus Number	0000_0000h	RW	page 64
D8h	LDT2 Type	page 65	RO	page 65
<b>Notes:</b> <ol style="list-style-type: none"> <li>1. The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.</li> <li>2. Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.</li> </ol>				

### 3.3.1 Device/Vendor ID Register

This register specifies the device and vendor IDs for the Function 0 registers and is part of the standard PCI configuration header.

#### Device/Vendor ID Register

Function 0: Offset 00h

31	16	15	0
DevID		VenID	

Bits	Mnemonic	Function	R/W	Reset
31–16	DevID	Device ID	R	1100h
15–0	VenID	Vendor ID	R	1022h

#### Field Descriptions

**Vendor ID (VenID)**—Bits 15–0. This read-only value is defined as 1022h for AMD.

**Device ID (DevID)**—Bits 31–16. This read-only value is defined as 1100h for the HyperTransport technology configuration function.

### 3.3.2 Status/Command Register

This register contains status and command information for the Function 0 registers and is part of the standard PCI configuration header.

#### Status/Command Register

Function 0: Offset 04h

31	16	15	0
Status			Command

Bits	Mnemonic	Function	R/W	Reset
31–16		Status	R	0010h
15–0		Command	R	0000h

#### Field Descriptions

**Command**—Bits 15–0. This read-only value is defined as 0000h.

**Status**—Bits 31–16. This read-only value is defined as 0010h to indicate that the processor has a capabilities list containing configuration information specific to HyperTransport technology.

### 3.3.3 Class Code/Revision ID Register

This register specifies the class code and revision for the Function 0 registers and is part of the standard PCI configuration header.

#### Class Code/Revision ID Register

Function 0: Offset 08h

31	24	23	16	15	8	7	0
BCC			SCC		PI		RevID

Bits	Mnemonic	Function	R/W	Reset
31–24	BCC	Base Class Code	R	06h
23–16	SCC	Subclass Code	R	00h
15–8	PI	Programming Interface	R	00h
7–0	RevID	Revision ID	R	00h

#### Field Descriptions

**Revision ID (RevID)**—Bits 7–0.

**Programming Interface (PI)**—Bits 15–8. This read-only value is defined as 00h.

**Sub Class Code (SCC)**—Bits 23–16. This read-only value is defined as 00h.

**Base Class Code (BCC)**—Bits 31–24. This read-only value is defined as 06h for a host bridge device.

### 3.3.4 Header Type Register

This register specifies the header type for the Function 0 registers and is part of the standard PCI configuration header.

#### Header Type Register

Function 0: Offset 0Ch

31	24	23	16	15	8	7	0	
BIST				HType		LatTimer		CLS

Bits	Mnemonic	Function	R/W	Reset
31–24	BIST	BIST	R	00h
23–16	HType	Header Type	R	80h
15–8	LatTimer	Latency Timer	R	00h
7–0	CLS	Cache Line Size	R	00h

#### Field Descriptions

**CacheLineSize (CLS)**—Bits 7–0. This read-only value is defined as 00h.

**LatencyTimer (LatTimer)**—Bits 15–8. This read-only value is defined as 00h.

**HeaderType (HType)**—Bits 23–16. This read-only value is defined as 80h to indicate that multiple functions are present in the configuration header and that the header layout corresponds to a device header as opposed to a bridge header. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**BIST**—Bits 31–24. This read-only value is defined as 00h.

### 3.3.5 Capabilities Pointer Register

This register contains a capabilities pointer to a linked capabilities list of registers specific to HyperTransport technology, with one set for each HyperTransport link. It is part of the standard PCI configuration header.

#### Capabilities Pointer Register

Function 0: Offset 34h

31	8	7	0
reserved			CapPtr

Bits	Mnemonic	Function	R/W	Reset
31–8	reserved		R	0
7–0	CapPtr	Capability Pointer	R	

## Field Descriptions

**Capability Pointer (CapPtr)**—Bits 7–0. Points to the first available HyperTransport technology capabilities block. Depending on the specific HyperTransport link configuration physically present on the processor, this will be either 80h (LDT0), A0h (LDT1) or C0h (LDT2).

## 3.3.6 Routing Table Node *i* Registers

The routing table contains three entries—one for requests RQRoute[7:0], one for responses RPRoute[7:0], and one for broadcasts BCRoute[7:0].

A maximum of eight nodes with three links per node is supported in an AMD Opteron™ system.

These table entries can be read and written, and its contents are not maintained through a warm reset. At reset, all table entries are initialized to the value 01h, indicating that packets should be accepted by this node. Care must be exercised when changing table entries after reset to ensure that connectivity is not lost during the process.

### 3.3.6.1 Request Routing Table

Each node contains a routing table for use with directed requests, with one entry for each of the eight possible destination Node IDs. The value in each entry indicates which outgoing link is used for request packets directed to that particular destination node (DestNode). A 1 in a given bit position indicates that the request is routed through the corresponding output link. Bit 0, when set to 1, indicates that the request must be accepted by this node.

### 3.3.6.2 Response Routing Table

Each node contains a routing table for use with responses, with one entry for each of the eight possible destination node IDs. The value in each entry indicates which outgoing link is used for response packets directed to that particular destination node (DestNode). A 1 in a given bit position indicates that the response is routed through the corresponding output link. Bit 0, when set to 1, indicates that the response must be accepted by this node.

### 3.3.6.3 Broadcast Routing Table

Each node contains a routing table for use with broadcast and probe requests with one entry for each of the eight possible source node IDs. Each entry contains a single bit for each of the outbound links from the node. The packet is forwarded on all links with their corresponding links set to a 1. Bit 0 when set to 1 indicates that the broadcast must be accepted by this node. The algorithm to generate the routing table follows the backbone first rule.

**Routing Table Node 0–7 Registers****Function 0: Offset 40h, 44h, 48h, 4Ch,  
50h, 54h, 58h, 5Ch**

31	20	19	16	15	12	11	8	7	4	3	0
reserved				BCRte	reserved		RPRte	reserved		RQRte	

Bits	Mnemonic	Function	R/W	Reset
31–20	reserved		R	000h
19–16	BCRte	Broadcast Route	R/W	1h
15–12	reserved		R	0h
11–8	RPRte	Response Route	R/W	1h
7–4	reserved		R	0h
3–0	RQRte	Request Route	R/W	1h

**Field Descriptions**

**Request Route (RQRte)**—Bits 3–0. The Request Route field defines the node or link to which a request packet is forwarded. Request packets are routed to only one destination and the table index is based on the destination Node ID.

Bit [0] = Route to this node

Bit [1] = Route to Link 0

Bit [2] = Route to Link 1

Bit [3] = Route to Link 2

**Response Route (RPRte)**—Bits 11–8. The Response Route field defines the node or link to which a response packet is forwarded. Response packets are routed to only one destination and the table index is based on the destination Node ID.

Bit [0] = Route to this node

Bit [1] = Route to Link 0

Bit [2] = Route to Link 1

Bit [3] = Route to Link 2

**Broadcast Route (BCRte)**—Bits 19–16. The Broadcast Route field defines the node or link(s) to which a broadcast packet is forwarded. Broadcasts may be routed to more than one destination. The Node ID that indexes into the table is the source Node ID.

Bit [0] = Route to this node

Bit [1] = Route to Link 0

Bit [2] = Route to Link 1

Bit [3] = Route to Link 2

### 3.3.7 Node ID Register

The Node ID register contains node ID, node count and CPU core count information (a node can have more than one CPU core).

It is expected that system configuration software will program the Node ID as part of coherent HyperTransport link configuration. Correct system operation depends on an assignment of distinct node ID values not exceeding NodeCnt[2:0]. For example, the Node IDs in a 4-node system must be 0, 1, 2, and 3; an example of an incorrect Node ID assignment in this system is 0, 1, 3, and 4. The Node ID will also be used as the initial APIC ID for the local APIC.

#### Node ID Register

#### Function 0: Offset 60h

31	20	19	16	15	14	12	11	10	8	7	6	4	3	2	0
reserved				CpuCnt	reserved	LkNode		reserved	SbNode		reserved	NodeCnt		reserved	NodeId

Bits	Mnemonic	Function	R/W	Reset
31–20	reserved		R	0
19–16	CPUCnt	CPU Count	R/W	0
15	reserved		R	0
14–12	LkNode	Lock Controller Node ID	R/W	0
11	reserved		R	0
10–8	SbNode	HyperTransport I/O Hub Node ID	R/W	0
7	reserved		R	0
6–4	NodeCnt	Node Count	R/W	0
3	reserved		R	0
2–0	NodeID	This Node ID	R/W	X

#### Field Descriptions

**Node ID (NodeID)**—Bits 2–0. Defines the Node ID of this node. It resets to 0 for the boot strap processor (BSP) and to 7h for all other nodes.

**Node Count (NodeCnt)**—Bits 6–4. Specifies the number of coherent nodes in the system. Note that the hardware allows only values to be programmed into this field that are consistent with the multiprocessor capabilities of the device, as specified in “Northbridge Capabilities Register” on page 165. This field will not be updated if there is an attempt to write a values that is inconsistent with the specified multiprocessor capabilities.

000b = 1 node  
 001b = 2 nodes  
 010b = 3 nodes  
 011b = 4 nodes  
 101b = 6 nodes

111b = 8 nodes

**HyperTransport I/O Hub Node ID (SbNode)**—Bits 10–8. Defines the Node ID of the node that owns the HyperTransport link to the HyperTransport I/O hub.

**Lock Controller Node ID (LkNode)**—Bits 14–12. Defines the Node ID of the node that contains the Lock Controller. The lock controller node (LkNode) must be the same as the HyperTransport I/O hub node (SbNode).

**CPU Count (CPUCnt)**—Bits 19–16. Defines the number of CPU cores in the system.

0000b = 1 CPU

...

1111b = 16 CPUs

### 3.3.8 Unit ID Register

The Unit ID register specifies the Unit ID of each functional unit on the processor. Under normal operation, there is no need for software to change the default Unit ID assignments. This register also contains a pointer to the HyperTransport I/O hub link.

#### Unit ID Register

#### Function 0: Offset 64h

31	10	9	8	7	6	5	4	3	2	1	0	
reserved								SbLink	HbUnit	McUnit	C1Unit	C0Unit

Bits	Mnemonic	Function	R/W	Reset
31–10	reserved		R	0
9–8	SbLink	HyperTransport I/O Hub Link ID	R/W	00b
7–6	HbUnit	Host Bridge Unit ID	R/W	11b
5–4	McUnit	Memory Controller Unit ID	R/W	10b
3–2	C1Unit	CPU1 Unit ID	R/W	01b
1–0	C0Unit	CPU0 Unit ID	R/W	00b

#### Field Descriptions

**CPU0 Unit ID (C0Unit)**—Bits 1–0. Defines the Unit ID of CPU0 core.

**CPU1 Unit ID (C1Unit)**—Bits 3–2. Defines the Unit ID of CPU1 core, if it is implemented.

**Memory Controller Unit ID (McUnit)**—Bits 5–4. Defines the Unit ID of the memory controller.

**Host Bridge Unit ID (HbUnit)**—Bits 7–6. Defines the Unit ID of the host bridge. The host bridge is used to bridge between coherent and noncoherent HyperTransport technology domains.

**HyperTransport I/O Hub Link ID (SbLink)**—Bits 9–8. Defines the link to which the HyperTransport I/O hub is connected. It is only used by the node which owns the HyperTransport I/O hub, as indicated in the Node ID register.

- 00b = LDT0 (default)
- 01b = LDT1
- 10b = LDT2
- 11b = Undefined

### 3.3.9 HyperTransport™ Transaction Control Register

The HyperTransport Transaction Control register configures the high-level HyperTransport protocols. It can be used to optimize system performance or change HyperTransport technology transaction protocols.

#### HyperTransport™ Transaction Control Register

**Function 0: Offset 68h**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DisIsocWrMedPri	DisIsocWrHiPri	DisWrLoPri	EnCpuRdHiPri	HiPriBypCnt	MedPriBypCnt	reserved		DsNpReqLmt	SeqIdSrcNodeEn	ApicExtSpur	ApicExtId	ApicExtBrdCst	LintEn	LimitCldtCfg	BufRelPri	ChglIsocToOrd	RspPassPW	DisFillp	DisRmtPMemC	DisPMemC	CPUReqRspPassPW	CPUReqPassPW	Cpu1En	DisMTS	DisWrDwP	DisWrBP	DisRdDwP	DisRdBP			

Bits	Mnemonic	Function	R/W	Reset
31	DisIsocWrMedPri	Disable medium priority isochronous writes	R/W	0
30	DisIsocWrHiPri	Disable high priority isochronous writes	R/W	0
29	DisWrLoPri	Disable low priority writes	R/W	0
28	EnCpuRdHiPri	Enable high priority CPU reads	R/W	0
27–26	HiPriBypCnt	High-priority bypass count	R/W	11b
25–24	MedPriBypCnt	Medium-priority bypass count	R/W	11b
23	reserved		R	0
22–21	DsNpReqLmt	Downstream non-posted request limit	R/W	00b
20	SeqIdSrcNodeEn	Sequence ID source node enable	R/W	0
19	ApicExtSpur	APIC extended spurious vector enable	R/W	0
18	ApicExtId	APIC extended ID enable	R/W	0
17	ApicExtBrdCst	APIC extended broadcast enable	R/W	0
16	LintEn	Local interrupt conversion enable	R/W	0
15	LimitCldtCfg	Limit coherent HyperTransport configuration space range	R/W	0
14–13	BufRelPri	Buffer release priority select	R/W	00b
12	ChgIsocToOrd	Change ISOC to Ordered	R/W	0
11	RspPassPW	Response PassPW	R/W	0
10	DisFillP	Disable fill probe	R/W	0



Bits	Mnemonic	Function	R/W	Reset
9	DisRmtPMemC	Disable remote probe memory cancel	R/W	0
8	DisPMemC	Disable probe memory cancel	R/W	0
7	CPU RdRspPassPW	CPU Read response PassPW	R/W	0
6	CPU ReqPassPW	CPU request PassPW	R/W	0
5	Cpu1En	CPU1 enable	R/W	0
4	DisMTS	Disable memory controller target start	R/W	0
3	DisWrDwP	Disable write doubleword probes	R/W	0
2	DisWrBP	Disable write byte probes	R/W	0
1	DisRdDwP	Disable read doubleword probe	R/W	0
0	DisRdBP	Disable read byte probe	R/W	0

## Field Descriptions

**Disable Read Byte Probe (DisRdBP)**—Bit 0. This bit determines if probes are issued for CPU-generated RdSized byte (must be 0 for multi-CPU core systems; recommended to be 1 for uniprocessor, single CPU core systems)

- 0 = Probes issued
- 1 = Probes not issued

**Disable Read Doubleword Probe (DisRdDwP)**—Bit 1. This bit determines if probes are issued for CPU-generated RdSized Doubleword (must be 0 for multi-CPU core systems; recommended to be 1 for uniprocessor, single CPU core systems)

- 0 = Probes issued
- 1 = Probes not issued

**Disable Write Byte Probes (DisWrBP)**—Bit 2. This bit determines if probes are issued for CPU-generated WrSized byte (must be 0 for multi-CPU core systems; recommended to be 1 for uniprocessor, single CPU core systems)

- 0 = Probes issued
- 1 = Probes not issued

**Disable Write Doubleword Probes (DisWrDwP)**—Bit 3. This bit determines if probes are issued for CPU-generated WrSized Doubleword (must be 0 for multi-CPU core systems; recommended to be 1 for uniprocessor, single CPU core systems)

- 0 = Probes issued
- 1 = Probes not issued

**Disable Memory Controller Target Start (DisMTS)**—Bit 4. Disables use of TgtStart. TgtStart is used to improve scheduling of back-to-back ordered transactions by indicating when the first transaction is received and ordered at the memory controller.

- 0 = TgtStart packets are generated
- 1 = TgtStart packets are not generated

**CPU1 Enable (Cpu1En)**—Bit 5. Enables a second CPU core, if it is implemented on the node. When this bit is set the second CPU core begins fetching code. This bit should only be set for nodes which have two CPU cores.

- 0 = Second CPU core disabled or not present
- 1 = Second CPU core enabled

**CPU Request PassPW (CPUReqPassPW)**—Bit 6. Allows CPU-generated requests to pass posted writes.

- 0 = CPU requests do not pass posted writes
- 1 = CPU requests pass posted writes

**CPU Read Response PassPW (CPURdRspPassPW)**—Bit 7. Allows RdResponses to CPU-generated reads to pass posted writes.

- 0 = CPU responses do not pass posted writes
- 1 = CPU responses pass posted writes

**Disable Probe Memory Cancel (DisPMemC)**—Bit 8. Disables generation of the MemCancel command when a probe hits a dirty cache block. MemCancels are used to attempt to save DRAM and/or HyperTransport technology bandwidth associated with the transfer of stale DRAM data.

- 0 = Probes may generate MemCancels
- 1 = Probes may not generate MemCancels

**Disable Remote Probe Memory Cancel (DisRmtPMemC)**—Bit 9. Disables generation of the MemCancel command when a probe hits a dirty cache block unless the probed cache is on the same node as the memory controller. MemCancels are used to attempt to save DRAM and/or HyperTransport technology bandwidth associated with the transfer of stale DRAM data.

- 0 = Probes hitting dirty blocks generate memory cancel packets, regardless of the location of the probed cache
- 1 = Only probed caches on the same node as the target memory controller generate memory cancel packets

**Disable Fill Probe (DisFillP)**—Bit 10. Disables probes for CPU-generated fills (must be 0 for multiprocessor system; recommended to be 1 for uniprocessor system).

- 0 = Probes issued for cache fills
- 1 = Probes not issued for cache fills

**Response PassPW (RspPassPW)**—Bit 11. Causes the host bridge to set the PassPW bit in all downstream responses.

This technically breaks the PCI ordering rules but it is not expected to be an issue in the downstream direction. Setting this bit improves the latency of upstream requests by allowing the downstream responses to pass posted writes.

- 0 = Downstream response PassPW based on original request
- 1 = Downstream response PassPW set to 1

**Change ISOC to Ordered (ChgIsocToOrd)**—Bit 12. Causes bit 1 of RdSz/WrSz commands to be treated as an ordered bit in coherent HyperTransport technology, instead of as an isochronous bit. Setting this bit will disable prioritization of isochronous requests, but will enable tracking of ordered commands across coherent HyperTransport links. This bit should only be set if the performance of non-ordered peer-to-peer traffic across coherent HyperTransport links is optimized, or to disable isochronous prioritization.

- 0 = Bit 1 of coherent HyperTransport technology sized command used for isochronous prioritization
- 1 = Bit 1 of coherent HyperTransport technology sized command used for ordering

**Buffer Release Priority Select (BufRelPri)**—Bits 14–13. Selects the number of HyperTransport technology doublewords sent with a buffer release pending before the buffer release is inserted into the command/data stream of a busy link.

- 00b = 64 (default)
- 01b = 16
- 10b = 8
- 11b = 2

This should be set to a value of 10 (8 HyperTransport packets) to maximize HyperTransport technology bandwidth.

**Limit Coherent HyperTransport Configuration Space Range (LimitCldtCfg)**—Bit 15. Limits the extent of the coherent HyperTransport technology configuration space based on the number of nodes in the system. When this bit is set, configuration accesses that normally map to coherent HyperTransport space will be sent to noncoherent HyperTransport links instead, if these accesses attempt to access a non-existent node, as specified through the node count in the Node ID register. This bit should be set by BIOS once coherent HyperTransport fabric initialization is complete. Failure to do so will result in PCI configuration accesses to non-existent nodes being sent into the coherent HyperTransport routing fabric, causing the system to hang.

- 0 = No coherent HyperTransport configuration space restrictions
- 1 = Limit coherent HyperTransport configuration space based on number of nodes

**Local Interrupt Conversion Enable (LintEn)**—Bit 16. Enables the conversion of broadcast ExtInt/NMI HyperTransport technology interrupts to LINT0/1 before delivering to the local APIC. This conversion only takes place if the local APIC is hardware enabled.

- 0 = ExtInt/NMI interrupts unaffected
- 1 = ExtInt/NMI broadcast interrupts converted to LINT0/1

**APIC Extended Broadcast Enable (ApicExtBrdCst)**—Bit 17. Enables extended APIC broadcast functionality.

- 0 = APIC broadcast is 0Fh
- 1 = APIC broadcast is FFh

**APIC Extended ID Enable (ApicExtId)**—Bit 18. Enables extended APIC ID functionality.

- 0 = APIC ID is 4 bits
- 1 = APIC ID is 8 bits

**APIC Extended Spurious Vector Enable (ApicExtSpur)**—Bit 19. Enables extended APIC spurious vector functionality.

- 0 = Lower 4 bits of spurious vector are read-only 1111b
- 1 = Lower 4 bits of spurious vector are writable

**Sequence ID Source Node Enable (SeqIdSrcNodeEn)**—Bit 20. Causes the source node of requests to be used in the SeqID field of downstream noncoherent HyperTransport technology request packets. This may be desirable for some debug HyperTransport packet tracing applications, to match downstream packets with their originating CPU. For normal operation, this bit should be cleared, since this use of SeqID for source node determination may cause problems with packet ordering.

**Downstream non-posted request limit (DsNpReqLmt)**—Bits 22–21. Sets the number of downstream non-posted requests issued by CPU(s) which may be outstanding on the noncoherent HyperTransport links attached to this node. Non-posted requests from CPU(s) will be throttled such that they do not exceed the programmed value in this field.

- 00b = No limit
- 01b = Limited to 1
- 10b = Limited to 4
- 11b = Limited to 8

**Medium-Priority Bypass Count (MedPriBypCnt)**—Bits 25–24. The maximum number of times a medium priority access can pass a low priority access before medium priority mode is disabled for one access.

**High-Priority Bypass Count (HiPriBypCnt)**—Bits 27–26. The maximum number of times a high priority access can pass a medium or low priority access before high priority mode is disabled for one access.

**Enable High Priority CPU Reads (EnCpuRdHiPri)**—Bit 28. Enables CPU reads to be treated as high priority. CPU reads are treated as medium priority by default. Setting EnCpuRdHiPri changes their priority to high.

**Disable Low Priority Writes (DisWrLoPri)**—Bit 29. Disables non-isochronous writes from being treated as low priority. Non-isochronous writes are treated as low priority by default. Setting DisWrLoPri changes their priority to medium.

**Disable High Priority Isochronous Writes (DisIsocWrHiPri)**—Bit 30. Disables isochronous writes from being treated as high priority. Isochronous writes are treated as high priority by default. Setting DisIsocWrHiPri changes their priority to medium (if DisIsocWrMedPri is clear) or low (if DisIsocWrMedPri is set).

**Disable Medium Priority Isochronous Writes (DisIsocWrMedPri)**—Bit 31. Disables isochronous writes from being treated as medium priority. Isochronous writes are treated as high priority

by default. Setting DisIsocWrMedPri along with DisIsocWrHiPri changes their priority to low.

### 3.3.10 HyperTransport™ Initialization Control Register

The HyperTransport Initialization Control register controls the routing and request generation that follows device initialization. This register also contains a set of scratchpad read/write bits that are cleared by different initialization conditions and that may be used to distinguish between various types of initialization events. Note that the Request Disable and Routing Table Disable bits must not be cleared until the routing tables have been initialized.

## HyperTransport™ Initialization Control Register

### Function 0: Offset 6Ch

31	7	6	5	4	3	2	1	0
reserved					InitDet	BiosRstDet		ColdRstDet
					DefLnk		ReqDis	RouteTblDis

Bits	Mnemonic	Function	R/W	Reset
31–7	reserved		R	0
6	InitDet	INIT Detect	R/W	0
5	BiosRstDet	BIOS Reset Detect	R/W	0
4	ColdRstDet	Cold Reset Detect	R/W	0
3–2	DefLnk	Default Link	R	
1	ReqDis	Request Disable	R/W	
0	RouteTblDis	Routing Table Disable	R/W	1

## Field Descriptions

**Routing Table Disable (RouteTblDis)**—Bit 0. This bit determines whether the routing tables are used or whether default configuration access routing is used. It resets to 1, thereby routing requests to the Configuration Special Register (CSR) block and routing responses based on DefLnk. Once the routing tables have been set up this bit should be cleared.

- 0 = Packets are routed according to the routing tables  
1 = Packets are routed according to the default link field (DefLnk)

**Request Disable (ReqDis)**—Bit 1. This bit determines if the node is allowed to generate request packets. It resets to 0 for the BSP and to 1 for all other processors. This bit should be cleared by system initialization firmware once the system has been initialized from the BSP. This bit is set by hardware and cleared by software.

- 0 = Request packets may be generated  
1 = Request packets may not be generated

**Default Link (DefLnk)**—Bits 3–2. This field is used after a reset and before the routing tables are initialized. It is used by hardware to determine which link to route responses to and may be used by software as part of system connectivity discovery.

The default link field is loaded each time an incoming request is received, with the link ID of the link on which the packet arrived. It is read-only from software. It is only used to route packets during initialization, when the Routing Table Disable bit is set to 1, and only one outstanding request is active in the system at a time. During this interval, the value in the Default Link field is used to route responses, so that responses are always sent out on the link from which the last request was received. The register is updated as soon as a request is received, so reads of this register from the fabric return the link ID of the link on which the read command arrived. Reads from the CPU on the local node cause this field to get loaded with 11b.

00b = LDT0  
01b = LDT1  
10b = LDT2  
11b = CPU on same node

**Cold Reset Detect (ColdRstDet)**—Bit 4. This bit may be used to distinguish between a cold versus a warm reset event by setting the bit to 1 before an initialization event is generated. This bit is cleared by a cold reset but not by a warm reset.

**BIOS Reset Detect (BiosRstDet)**—Bit 5. This bit may be used to distinguish between a reset event generated by the BIOS versus a reset event generated for any other reason by setting the bit to 1 before initiating a BIOS-generated reset event. This bit is cleared by a cold reset but not by a warm reset.

**INIT Detect (InitDet)**—Bit 6. This bit may be used to distinguish between an INIT and a warm/cold reset by setting the bit to 1 before an initialization event is generated. This bit is cleared by a warm or cold reset but not by an INIT.

### 3.3.11 LDT<sub>i</sub> Capabilities Register

These registers define the capabilities of the LDT links. They also contain a capabilities pointer that points to the next item in the linked capabilities list.

#### LDT0, LDT1, LDT2 Capabilities Registers

Function 0: Offset 80h, A0h, C0h

31	29	28	27	26	25	24	23	22	18	17	16	15	8	7	0
CapType	DropOnUnInit	InbndEocErr	ActAsSlave	reserved	HostHide	ChainSide		DevNum	DblEnded	WarmReset		CapPtr		CapID	

Bits	Mnemonic	Function	R/W	Reset
31–29	CapType	Capability Type	R	001b
28	DropOnUnInit	Drop on Uninitialized Link	R	0
27	InbndEocErr	Inbound End-of-Chain Error	R	0
26	ActAsSlave	Act As Slave	R	0
25	reserved		R	0
24	HostHide	Host Hide	R	1
23	ChainSide	Chain Side	R	0
22–18	DevNum	Device Number	R	0
17	DblEnded	Double Ended	R	0
16	WarmReset	Warm Reset	R	1
15–8	CapPtr	Capability Pointer	R	
7–0	CapID	Capability ID (Always Reads 08h)	R	08h

## Field Descriptions

**Capability ID (CapID)**—Bits 7–0. A capability ID of 08h indicates HyperTransport technology capability.

**Capability Pointer (CapPtr)**—Bits 15–8. Contains a pointer to the next capability in the list. Depending on the specific HyperTransport link configuration physically present on the processor, this will be either A0h (LDT1), C0h (LDT2), or 00h, if it is the last one.

**Warm Reset (WarmReset)**—Bit 16. Allows a reset sequence initiated by the HyperTransport technology control register to be either warm or cold. Since resets initiated through the HyperTransport control register are not supported, this field is read-only 1.

**Double Ended (DblEnded)**—Bit 17. Indicates that there is another bridge at the far end of the noncoherent HyperTransport chain. Since double-hosted chains are not supported, this field is read-only 0.

**Device Number (DevNum)**—Bits 22–18. The device number of configuration accesses which Northbridge responds to when accessed from the noncoherent HyperTransport technology chain. Since double-hosted chains are not supported, this field is read-only 0.

**Chain Side (ChainSide)**—Bit 23. This bit indicates which side of the host bridge is being accessed. Since double-hosted chains are not supported, this bit is read-only 0.

**Host Hide (HostHide)**—Bit 24. This bit causes the memory system configuration space to be inaccessible from the noncoherent HyperTransport technology chain. Since double-hosted chains are not supported, this bit is read-only 1.

**Act As Slave (ActAsSlave)**—Bit 26. Since this function is not currently implemented, this bit is read-only 0.

**Inbound End-of-Chain Error (InbndEocErr)**—Bit 27. Since this function is not currently implemented, this bit is read-only 0.

**Drop on Uninitialized Link (DropOnUnInit)**—Bit 28. Since this function is not currently implemented, this bit is read-only 0.

**Capability Type (CapType)**—Bits 31–29. The code 001b designates a host interface HyperTransport technology capability.

### 3.3.12 LDTi Link Control Registers

These registers control LDT link operation and log link errors.

#### LDT0, LDT1, LDT2 Link Control Registers

**Function 0: Offset 84h, A4h, C4h**

31	30	28	27	26	24	23	22	20	19	18	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DwFcOutEn	WidthOut	DwFcInEn	WidthIn	DwFcOut	MaxWidthOut	DwFcIn	MaxWidthIn	LdtStopTriClkOvr	ExtCTL	LdtStopTriEn	IsocEn	LdtStopRcvDis	reserved	CrcErr	TransOff	RcvOff	InitComplete	LinkFail	CrcForceErr	CrcStartTest	CrcFloodEn	reserved					

Bits	Mnemonic	Function	R/W	Reset
31	DwFcOutEn	Doubleword Flow Control Out Enable	R	0
30–28	WidthOut	Link Width Out	R/W	0
27	DwFcInEn	Doubleword Flow Control In Enable	R	0
26–24	WidthIn	Link Width In	R/W	0
23	DwFcOut	Doubleword Flow Control Out	R	0
22–20	MaxWidthOut	Max Link Width Out	R	001b
19	DwFcIn	Doubleword Flow Control In	R	0
18–16	MaxWidthIn	Max Link Width In	R	001b
15	LdtStopTriClkOvr	HyperTransport Stop Tristate Clock Override	R/W	0
14	ExtCTL	Extended CTL Time	R/W	0
13	LdtStopTriEn	HyperTransport Stop Tristate Enable	R/W	0
12	IsocEn	Isochronous Enable	R	0
11	LdtStopRcvDis	HyperTransport Stop Receiver Disable	R/W	0
10	reserved		R	0
9–8	CrcErr	CRC_Error	R/W	0
7	TransOff	Transmitter Off	R/W	0
6	RcvOff	Receiver Off	R/W	0
5	InitComplete	Initialization Complete	R	0
4	LinkFail	Link Failure	R/W	0
3	CrcForceErr	CRC Force Error	R/W	0
2	CrcStartTest	CRC Start Test	R	0
1	CrcFloodEn	CRC Flood Enable	R/W	0
0	reserved		R	0



## Field Descriptions

**CRC Flood Enable (CrcFloodEn)**—Bit 1. If cleared to 0, this bit prevents CRC errors both from generating sync packets and causing the system to come down, and from setting the LinkFail bit. However, CRC checking logic still runs on all lanes enabled by LinkWidthIn, and detected errors still set the CRC Error bits to 1. Its reset value is 0.

- 0 = Do not generate sync packets on CRC error (default)
- 1 = Generate sync packets on CRC error

**CRC Start Test (CrcStartTest)**—Bit 2. Since this function is not currently implemented, this bit is read-only 0.

**CRC Force Error (CrcForceErr)**—Bit 3. When this bit is set, a bad CRC is generated on all transmitting lanes as enabled by LinkWidthOut. The covered data is not affected. This bit is readable and writable by software and resets to 0.

- 0 = Do not generate a bad CRC (default)
- 1 = Generate a bad CRC

**Link Failure (LinkFail)**—Bit 4. The LinkFail bit is set to 1 when a CRC error or a sync packet is detected after link initialization or if the link is not connected. See “Machine Check Architecture (MCA) Registers” on page 121 for more details. This bit is maintained through a warm reset and is cleared to 0 on a cold reset. LinkFail is set to 1 by hardware in the event of a link error that results in a sync flood. If this bit is set after a warm reset, BIOS must attempt to clear the bit by writing a 1 to determine if the link is not connected or if a sync flood occurred. If the BIOS can clear the bit, a sync flood occurred. This bit becomes valid after Function 0, Offset 98h, B8h, or D8h, LinkConPend bit is cleared by hardware.

- 0 = No link failure detected
- 1 = Link failure detected

**Initialization Complete (InitComplete)**—Bit 5. This read-only bit is reset to 0, and set to 1 by hardware when low-level link initialization is successfully complete. If there is no device on the other end of the link, or if that device is unable to properly perform the low-level link initialization protocol, the bit is not set to 1. Software must not attempt to generate any packets across the link until this bit is set to 1. CRC checking in the hardware does not begin until initialization is complete.

- 0 = Initialization not complete
- 1 = Initialization is complete

**Receiver Off (RcvOff)**—Bit 6. This bit disables the receiver from accepting any further packets. This bit resets to 0, and is set by software writing a 1 to the bit. This bit cannot be cleared by software—a write of 0 to this bit position has no effect. If the HyperTransport link is not connected, then this bit is set by hardware.

- 0 = Receiver on
- 1 = Receiver off

**Transmitter Off (TransOff)**—Bit 7. This bit provides a mechanism to shut off a link transmitter for power savings or EMI reduction. When set to 1, no output signals toggle on the link. This bit resets to 0, and is set by software writing a 1 to the bit. This bit cannot be cleared by software—a write of 0 to this bit position has no effect. If the HyperTransport link is not connected, then this bit is set by hardware.

- 0 = Transmitter on
- 1 = Transmitter off

**CRC\_Error (CrcErr)**—Bits 9–8. These bits are set to 1 by hardware when a CRC error is detected on an incoming link. Errors are detected and reported on a per byte lane basis where bit 8 corresponds to the least significant byte lane. Two bits are required to cover the maximum HyperTransport link width of 16 bits. Error bits for unimplemented (as specified by MaxWidthIn) or unused byte lanes return 0 when read.

These bits are maintained through a warm reset and is cleared to 0 on a cold reset. They are readable from software and are individually cleared to 0 by software by writing a 1.

- 00b = No error
- Bit [0] = 1 Error on Byte Lane 0
- Bit [1] = 1 Error on Byte Lane 1

**HyperTransport Stop Receiver Disable (LdtStopRcvDis)**—Bit 11. This bit enables power savings by controlling the state of the HyperTransport receiver. When this bit is set for any HyperTransport link, the HyperTransport receiver is disabled when LDTSTOP\_L is asserted. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Isochronous Enable (IsocEn)**—Bit 12. Since this function is not currently implemented, this bit is read-only 0.

**HyperTransport Stop Tristate Enable (LdtStopTriEn)**—Bit 13. This bit controls whether the transmitter tristates the link during the disconnected state of an LDTSTOP\_L sequence. When the bit is set the transmitter tristates the link. When the bit is clear, it continues to drive the link. For revision E when this bit is set and LdtStopTriClkOvr is clear, the delay specified in the ReConDel (Function 3, Offset D4h) field is applied before restarting the link. The bit is cleared by a cold reset and its value is maintained through a warm reset.

- 0 = Driven during disconnect of an LDTSTOP\_L
- 1 = Tristated during disconnect of an LDTSTOP\_L

**Extended CTL Time (ExtCTL)**—Bit 14. When this bit is set, CTL will be asserted for 50μs instead of 16 bit times during the link initialization sequence. This bit should be set if extended CTL is required by the device at the other end of the HyperTransport link, as indicated by the Extended CTL Required bit in the Feature Capability register. The bit is cleared by cold reset and its value is maintained through warm reset.

**HyperTransport Stop Tristate Clock Override (LdtStopTriClkOvr)**—Bit 15. This bit prevents the HyperTransport clock from being tristated when LDTSTOP\_L is asserted and

LdtStopTriEn is set. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Max Link Width In (MaxWidthIn)**—Bits 18–16. This field contains three bits that indicate the physical width of the incoming side of the HyperTransport link implemented by this device. For the AMD Athlon™ 64 and AMD Opteron™ Processors, this field is set to 000 or 001 to indicate an 8-bit or 16-bit link.

**Doubleword Flow Control In (DwFcIn)**—Bit 19. This bit is read-only 0 to indicate that this link does not support doubleword flow control.

**Max Link Width Out (MaxWidthOut)**—Bits 22–20. This field contains three bits that indicate the physical width of the outgoing side of the HyperTransport link implemented by this device. It is read-only. For the AMD Athlon™ 64 and AMD Opteron™ Processors, this field is set to 000 or 001 to indicate an 8-bit or 16-bit link.

**Doubleword Flow Control Out (DwFcOut)**—Bit 23. This bit is read-only 0 to indicate that this link does not support doubleword flow control.

**Link Width In (WidthIn)**—Bits 26–24. This field is similar to the WidthOut field, except that it controls the utilized width of the incoming side of the links implemented by this device.

The hardware will only allow values to be programmed into this field which are consistent with the width capabilities of the link as specified by MaxWidthIn. Attempts to write values inconsistent with the capabilities will result in this field not being updated.

**Doubleword Flow Control In Enable (DwFcInEn)**—Bit 27. Since this function is not currently implemented, this bit is read-only 0.

**Link Width Out (WidthOut)**—Bits 30–28. This field controls the utilized width (which may not exceed the physical width) of the outgoing side of the HyperTransport link. Software can read/write this field, and its value is maintained through a warm reset. After cold reset, this field is initialized by hardware based on the results of the link width negotiation sequence described in the HyperTransport technology specification. Based on sizing the devices at both ends of the link, software may then write a different value into the register. The chain must pass through warm reset or a LDTSTOP\_L disconnect sequence for the new width values to be reflected on the link.

The WidthOut CSR in the link transmitter must match the WidthIn CSR in the link receiver of the device on the other side of the link. The LinkWidthIn and LinkWidthOut registers within the same device are not required to have matching values.

The hardware will only allow values to be programmed into this field which are consistent with the width capabilities of the link as specified by MaxWidthOut. Attempts to write values inconsistent with the capabilities will result in this field not being updated

000b = 8-bit

001b = 16-bit

010b = reserved

011b = 32-bit  
100b = 2-bit  
101b = 4-bit  
110b = reserved  
111b = Link physically not connected

**Doubleword Flow Control Out Enable (DwFcOutEn)**—Bit 31. Since this function is not currently implemented, this bit is read-only 0.

### 3.3.13 LDTi Frequency/Revision Registers

These registers control the link frequency, specify the link frequency capabilities, and describe the HyperTransport technology specification level to which the specific link conforms.

#### LDT0, LDT1, LDT2 Frequency/Revision Registers      Function 0: Offset 88h, A8h, C8h

31	16	15	12	11	8	7	5	4	0
LnkFreqCap				Error	Freq	MajRev	MinRev		

Bits	Mnemonic	Function	R/W	Reset
31–16	LnkFreqCap	Link Frequency Capability	R	
15–12	Error	Error	R	0
11–8	Freq	Link Frequency	R/W	0
7–5	MajRev	Major Revision	R	001b
4–0	MinRev	Minor Revision	R	00010b

#### Field Descriptions

**Minor Revision (MinRev)**—Bits 4–0. Contains the minor revision of the HyperTransport technology specification to which the device conforms.

**Major Revision (MajRev)**—Bits 7–5. Contains the major revision of the HyperTransport technology specification level to which the device conforms.

**Link Frequency (Freq)**—Bits 11–8. Specifies the maximum operating frequency of the link's transmitter clock. The encoding of this field is shown below.

The LinkFreq is cleared by cold reset. Software can write a nonzero value to this register, and that value takes effect on the next warm reset or LDTSTOP\_L disconnect sequence.

The hardware will only allow values to be programmed into this field which are consistent with the frequency capabilities of the link as specified by LnkFreqCap. Attempts to write values inconsistent with the capabilities will result in this field not being updated.

It is possible to program this field for a higher frequency than the maximum allowed by the processor. Refer to the processor data sheet for the maximum operating frequency allowed for a given processor implementation.

Processors with multiple HyperTransport™ links are capable of operating the links at different frequencies. There are three supported link frequency groups:

800 MHz, 400 MHz, 200 MHz

1000 MHz

600 MHz

Processors can be configured with link frequencies from up to two link frequency groups.

0000b = 200 MHz

0001b = reserved

0010b = 400 MHz

0011b = reserved

0100b = 600 MHz

0101b = 800 MHz

0110b = 1000 MHz

0111b = reserved

1000–1110b = reserved

1111b = 100 MHz

If a link is not connected and the 200MHz link frequency is not used, then this field should be written to the frequency of one of the connected links.

**Note:** *The processor's HyperTransport links operate in Synchronous (Sync) mode as described in the HyperTransport I/O Link Specification. Sync mode operation requires the transmit and receive clocks for all links to be derived from the same time base. If different clock generators are used to drive the reference clock to the devices on both sides of the link, then the following requirements must be satisfied to ensure proper link operation:*

1. *All clock generators must use the same reference clock source.*
2. *Spread spectrum clocking must not be enabled in any of the clock generators unless the frequency deviations are synchronized between the outputs of all clock generators.*

**Error (Error)**—Bits 15–12. This function is not currently implemented.

**Link Frequency Capability (LnkFreqCap)**—Bits 31–16. Indicates the clock frequency capabilities of the link. Each bit corresponds to one of the 16 possible link frequency encodings.

While the frequency capabilities of different AMD Athlon™ 64 and AMD Opteron™ Processors may vary, the 200-MHz and 100-MHz frequencies are supported by the design.

### 3.3.14 LDT/ Feature Capability Registers

These registers identify features that are supported by the specific link.

## LDT0, LDT1, LDT2 Feature Capability Registers

## Function 0: Offset 8Ch, ACh, CCh



Bits	Mnemonic	Function	R/W	Reset
31–9	reserved		R	0
8	ExtRegSet	Extended Register Set	R	0
7–4	reserved		R	0
3	ExtCTLRqd	Extended CTL Required	R	0
2	CrcTstMode	CRC Test Mode	R	0
1	LdtStopMode	HyperTransport Stop Mode	R	1
0	IsocMode	Isochronous Flow Control Mode	R	0

### Field Descriptions

**Isochronous Flow Control Mode (IsocMode)**—Bit 0. This is a read-only bit that reflects whether isochronous flow control operation is supported. This bit is set to 0, indicating no support for this feature.

**HyperTransport Stop Mode (LdtStopMode)**—Bit 1. This is a read-only bit that indicates whether the LDTSTOP\_L protocol is supported. This bit is set to 1, indicating support for this feature.

**CRC Test Mode (CrcTstMode)**—Bit 2. This is a read-only bit that indicates whether CRC test mode is supported. This bit is set to 0, indicating no support for this feature.

**Extended CTL Required (ExtCTLRqd)**—Bit 3. This is a read-only bit that indicates whether extended CTL is required. This bit is set to 0, indicating that extended CTL is not required.

**Extended Register Set (ExtRegSet)**—Bit 8. This is a read-only bit that indicates whether the Enumeration Scratchpad, Error Handling and Memory Base/Limit Upper registers are supported. This bit is set to 0, indicating no support for this feature.

### 3.3.15 LDTi Buffer Count Registers

These registers specify the number of command and data buffers for each virtual channel available for use by the transmitter at the other end of the specific link. See “XBAR Flow Control Buffers” on page 143 for more information on command and data buffers.

**Note:** The reset values for each of the LDTn Buffer Count registers depend on the link connection type (coherent HyperTransport or noncoherent HyperTransport technology). Because hardware attempts to choose optimal settings, this register should not, in general, need to be changed.

**LDT0, LDT1, LDT2 Buffer Count Registers****Function 0: Offset 90h, B0h, D0h**

31	27	26	24	23	22	20	19	18	16	15	12	11	8	7	4	3	0
reserved		RspD		reserved	PReqD		reserved	ReqD		Probe		Rsp		PReq		Req	

Bits	Mnemonic	Function	R/W	Coherent HyperTransport Reset	Noncoherent HyperTransport Reset
31–27	reserved		R	0	0
26–24	RspD	Response Data Buffer Count	R/W	100b	010b
23	reserved		R	0	0
22–20	PReqD	Posted Request Data Buffer Count	R/W	001b	101b
19	reserved		R	0	0
18–16	ReqD	Request Data Buffer Count	R/W	011b	001b
15–12	Probe	Probe Buffer Count	R/W	5h	0h
11–8	Rsp	Response Buffer Count	R/W	6h	4h
7–4	PReq	Posted Request Buffer Count	R/W	1h	5h
3–0	Req	Request Buffer Count	R/W	3h	6h

**Field Descriptions**

**Request Buffer Count (Req)**—Bits 3–0. Defines the number of request command buffers available for use by the transmitter at the other end of the link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Posted Request Buffer Count (PReq)**—Bits 7–4. Defines the number of Posted request command buffers available for use by the transmitter at the other end of the link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Response Buffer Count (Rsp)**—Bits 11–8. Defines the number of response buffers available for use by the transmitter at the other end of the link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Probe Buffer Count (Probe)**—Bits 15–12. Defines the number of probe buffers available for use by the transmitter at the other end of the link. This field must be 0 for a noncoherent HyperTransport link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Request Data Buffer Count (ReqD)**—Bits 18–16. Defines the number of request data buffers available for use by the transmitter at the other end of the link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Posted Request Data Buffer Count (PReqD)**—Bits 22–20. Defines the number of posted request data buffers available for use by the transmitter at the other end of the link. See the register

layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

**Response Data Buffer Count (RspD)**—Bits 26–24. Defines the number of response data buffers available for use by the transmitter at the other end of the link. See the register layout for the default reset settings for coherent and noncoherent HyperTransport technology links.

### 3.3.16 LDT/ Bus Number Registers

If the specific link (LDT0, LDT1, or LDT2) is noncoherent HyperTransport technology, this register specifies the bus numbers downstream (behind) the host bridge. If the link is coherent HyperTransport technology, this register has no meaning.

#### LDT0, LDT1, LDT2 Bus Number Registers

Function 0: Offset 94h, B4h, D4h

31	24	23	16	15	8	7	0
reserved				SubBusNum	SecBusNum	PriBusNum	

Bits	Mnemonic	Function	R/W	Reset
31–24	reserved		R	0
23–16	SubBusNum	Subordinate Bus Number	R/W	0
15–8	SecBusNum	Secondary Bus Number	R/W	0
7–0	PriBusNum	Primary Bus Number	R	0

#### Field Descriptions

**Primary Bus Number (PriBusNum)**—Bits 7–0. Defines the primary bus number. Because the primary bus is the coherent HyperTransport technology fabric, this field always reads 0.

**Secondary Bus Number (SecBusNum)**—Bits 15–8. Defines the secondary bus number.

The Secondary Bus Number register is used to record the bus number of the bus segment to which the secondary interface of the host bridge is connected. Configuration software programs the value in this register. If this link contains the HyperTransport I/O hub, the secondary bus number must be programmed to 0.

**Subordinate Bus Number (SubBusNum)**—Bits 23–16. Defines the subordinate bus number.

The Subordinate Bus Number register is used to record the bus number of the highest numbered bus segment that is behind (or subordinate to) the host bridge. Configuration software programs the value in this register.

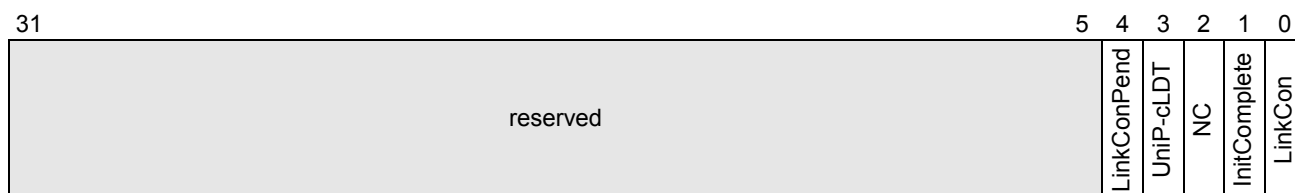


### 3.3.17 LDTi Type Registers

These registers designate the type of HyperTransport link attached to the specific link. The bits are set by hardware after link initialization.

#### LDT0, LDT1, LDT2 Type Registers

#### Function 0: Offset 98h, B8h, D8h



Bits	Mnemonic	Function	R/W	Reset
31–5	reserved		R	0
4	LinkConPend	Link Connect Pending	R	
3	UniP-cLDT	UniP-cLDT	R	
2	NC	Non Coherent	R	
1	InitComplete	Initialization Complete	R	
0	LinkCon	Link Connected	R	

#### Field Descriptions

**Link Connected (LinkCon)**—Bit 0. Indicates that the link is connected. It is valid once the LinkConPend bit is clear.

- 0 = Not connected
- 1 = Connected

**Initialization Complete (InitComplete)**—Bit 1. Set to 1 to indicate that the initialization of the link has completed. (It is a duplicate of Bit 5 in HyperTransport Link Control). The NC and UniP-cLDT bits are invalid until link initialization is complete.

- 0 = Initialization not complete
- 1 = Initialization is complete.

**Non Coherent (NC)**—Bit 2. Defines the link type (coherent versus noncoherent).

- 0 = Coherent HyperTransport technology
- 1 = Noncoherent HyperTransport technology

**UniP-cLDT (UniP-cLDT)**—Bit 3. Further qualifies the NC link type bit. A 1 indicates that this link is a uniprocessor coherent HyperTransport link connected to an external Northbridge.

- 0 = Normal coherent HyperTransport or noncoherent HyperTransport link
- 1 = Uniprocessor coherent HyperTransport link to external Northbridge

**Link Connect Pending (LinkConPend)**—Bit 4. Qualifies the LinkCon bit and is set to 1 when hardware is attempting to determine whether a link is connected or not.

0 = Link connection determination complete  
1 = Link connection still being determined

## 3.4 Function 1—Address Map

The address map defines the address spaces assigned to DRAM, memory-mapped I/O, PCI I/O, and configuration accesses and also specifies destination information for each to facilitate routing of each access to the appropriate target. Table 3 lists each Function 1 configuration register.

**Table 3. Function 1 Configuration Registers**

Offset	Register Name				Reset	Access	Description
00h	Device ID		Vendor ID (AMD)		1101_1022h	RO	page 41
04h	Status <sup>1</sup>		Command		0000_0000h	RO	
08h	Base Class Code	Subclass Code	Program Interface	Revision ID	0600_0000h	RO	page 69
0Ch	BIST	Header Type	Latency Timer	Cache Line Size	0080_0000	RO	page 69
10h	Base Address 0				0000_0000h	RO	
14h	Base Address 1				0000_0000h	RO	
18h	Base Address 2				0000_0000h	RO	
1Ch	Base Address 3				0000_0000h	RO	
20h	Base Address 4				0000_0000h	RO	
24h	Base Address 5				0000_0000h	RO	
28h	Card Bus CIS Pointer				0000_0000h	RO	
2Ch	Sub-System ID		Sub-System Vendor ID		0000_0000h	RO	
30h	ROM Base Address				0000_0000h	RO	
34h	Capabilities				0000_0000h	RO	
38h	reserved				0000_0000h	RO	
3Ch	Max Latency	Min GNT	Int Pin	Int Line	0000_0000h	RO	
40h	DRAM Base 0					RW	page 71
44h	DRAM Limit 0					RW	page 72
48h	DRAM Base 1					RW	page 71
4Ch	DRAM Limit 1					RW	page 72

**Notes:**

1. The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.
2. Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.

**Table 3. Function 1 Configuration Registers (Continued)**

Offset	Register Name	Reset	Access	Description
50h	DRAM Base 2		RW	page 71
54h	DRAM Limit 2		RW	page 72
58h	DRAM Base 3		RW	page 71
5Ch	DRAM Limit 3		RW	page 72
60h	DRAM Base 4		RW	page 71
64h	DRAM Limit 4		RW	page 72
68h	DRAM Base 5		RW	page 71
6Ch	DRAM Limit 5		RW	page 72
70h	DRAM Base 6		RW	page 71
74h	DRAM Limit 6		RW	page 72
78h	DRAM Base 7		RW	page 71
7Ch	DRAM Limit 7		RW	page 72
80h	Memory-Mapped I/O Base 0		RW	page 74
84h	Memory-Mapped I/O Limit 0		RW	page 75
88h	Memory-Mapped I/O Base 1		RW	page 74
8Ch	Memory-Mapped I/O Limit 1		RW	page 75
90h	Memory-Mapped I/O Base 2		RW	page 74
94h	Memory-Mapped I/O Limit 2		RW	page 75
98h	Memory-Mapped I/O Base 3		RW	page 74
9Ch	Memory-Mapped I/O Limit 3		RW	page 75
A0h	Memory-Mapped I/O Base 4		RW	page 74
A4h	Memory-Mapped I/O Limit 4		RW	page 75
A8h	Memory-Mapped I/O Base 5		RW	page 74
ACh	Memory-Mapped I/O Limit 5		RW	page 75
B0h	Memory-Mapped I/O Base 6		RW	page 74
B4h	Memory-Mapped I/O Limit 6		RW	page 75
B8h	Memory-Mapped I/O Base 7		RW	page 74
BCh	Memory-Mapped I/O Limit 7		RW	page 75
C0h	PCI I/O Base 0		RW	page 76
C4h	PCI I/O Limit 0		RW	page 77
C8h	PCI I/O Base 1		RW	page 76
CCh	PCI I/O Limit 1		RW	page 77
<b>Notes:</b> <ol style="list-style-type: none"> <li>1. The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.</li> <li>2. Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.</li> </ol>				

**Table 3. Function 1 Configuration Registers (Continued)**

Offset	Register Name	Reset	Access	Description
D0h	PCI I/O Base 2		RW	page 76
D4h	PCI I/O Limit 2		RW	page 77
D8h	PCI I/O Base 3		RW	page 76
DCh	PCI I/O Limit 3		RW	page 77
E0h	Configuration Base and Limit 0		RW	page 78
E4h	Configuration Base and Limit 1		RW	page 78
E8h	Configuration Base and Limit 2		RW	page 78
ECh	Configuration Base and Limit 3		RW	page 78
F0h	DRAM Hole Address	0000_0000	R/W	page 80
<b>Notes:</b> <ol style="list-style-type: none"> <li>1. The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.</li> <li>2. Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.</li> </ol>				

### 3.4.1 Device/Vendor ID Register

This register specifies the device and vendor IDs for the Function 1 registers and is part of the standard PCI configuration header.

#### Device/Vendor ID Register

**Function 1: Offset 00h**

31	16	15	0
Device ID		Vendor ID	

Bits	Mnemonic	Function	R/W	Reset
31–16	DevID	Device ID	R	1101h
15–0	VenID	Vendor ID	R	1022h

#### Field Descriptions

**Vendor ID (VenID)**—Bits 15–0. This read-only value is defined as 1022h for AMD.

**Device ID (DevID)**—Bits 31–16. This read-only value is defined as 1101h for the HyperTransport technology configuration function.

### 3.4.2 Class Code/Revision ID Register

This register specifies the class code and revision for the Function 1 registers and is part of the standard PCI configuration header.

#### Class Code/Revision ID Register

Function 1: Offset 08h

31	24	23	16	15	8	7	0
Base Class Code			Subclass Code		Programming Interface		Revision ID

Bits	Mnemonic	Function	R/W	Reset
31–24	BCC	Base Class Code	R	06h
23–16	SCC	Subclass Code	R	00h
15–8	PI	Programming Interface	R	00h
7–0	RevID	Revision ID	R	00h

#### Field Descriptions

**Revision ID (RevID)**—Bits 7–0.

**Programming Interface (PI)**—Bits 15–8. This read-only value is defined as 00h.

**Sub Class Code (SCC)**—Bits 23–16. This read-only value is defined as 00h.

**Base Class Code (BCC)**—Bits 31–24. This read-only value is defined as 06h for a host bridge device.

### 3.4.3 Header Type Register

This register specifies the header type for the Function 1 registers and is part of the standard PCI configuration header.

#### Header Type Register

Function 1: Offset 0Ch

31	24	23	16	15	8	7	0	
BIST				HType		LatTimer		CLS

Bits	Mnemonic	Function	R/W	Reset
31–24	BIST	BIST	R	00h
23–16	HType	Header Type	R	80h
15–8	LatTimer	Latency Timer	R	00h
7–0	CLS	Cache Line Size	R	00h

## Field Descriptions

**CacheLineSize (CLS)**—Bits 7–0. This read-only value is defined as 00h.

**LatencyTimer (LatTimer)**—Bits 15–8. This read-only value is defined as 00h.

**HeaderType (HType)**—Bits 23–16. This read-only value is defined as 80h to indicate that multiple functions are present in the configuration header and that the header layout corresponds to a device header as opposed to a bridge header. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**BIST**—Bits 31–24. This read-only value is defined as 00h.

### 3.4.4 DRAM Address Map

These registers define sections of the memory address map for which accesses should be routed to DRAM. DRAM regions must not overlap each other. For addresses within the specified range of a base/limit pair, requests are routed to the memory controller on the node specified by the destination Node ID.

System addresses are considered to be within the defined range if they are greater than or equal to the base and less than or equal to the limit. For the purposes of this comparison, the lower unspecified bits of the base are assumed to be 0s and the lower unspecified bits of the limit are assumed to be 1s.

An address that maps to both DRAM and memory-mapped I/O will be routed to MMIO.

Programming of the DRAM address maps must be consistent with the Top Of Memory and Memory Type Range registers (see Chapter 13, “Processor Configuration Registers”). Accesses from the CPU can only access the DRAM address maps if the corresponding CPU memory type is DRAM. For accesses from I/O devices, the lookup is based on address only.

Each base/limit set of DRAM address maps is associated with a particular Node ID (see “Node ID Register” on page 46). The DRAM Base/Limit 0 registers specify the DRAM attached to node 0. Similarly, DRAM on nodes 1–7 is described by base/limit registers 1–7. Note that the destination NodeId field must still be written to ensure correct operation.

When node interleaving is enabled, each node's DRAM limit must be set to the Top Of Memory and each node's DRAM base must be set to 0. The node to which an address is routed to when nodes are interleaved is defined by IntlvEn (Function 1, Offset 40h, 48h, etc.) and IntlvSel (Function 1, Offset 44h, 4Ch, etc.). Each node must be configured with the same amount of DRAM when node interleaving is enabled.

Address routed to the DRAM controller (InputAddr) is calculated from the system address (SysAddr) in the following way:

$\text{DramAddr}[39:0] = \{\text{SysAddr}[39:24] - \text{DRAMBase}[39:24], \text{SysAddr}[23:0]\},$

$\text{InputAddr}[35:0] = \{\text{DramAddr}[35:12], \text{DramAddr}[11:0]\}$  when node memory is not interleaved,

$\text{InputAddr}[35:0] = \{\text{DramAddr}[36:13], \text{DramAddr}[11:0]\}$  when 2 nodes are interleaved,

InputAddr[35:0] = {DramAddr[37:14], DramAddr[11:0]} when 4 nodes are interleaved,  
 InputAddr[35:0] = {DramAddr[38:15], DramAddr[11:0]} when 8 nodes are interleaved.

### 3.4.4.1 DRAM Base *i* Registers

#### DRAM Base 0–7 Registers      Function 1: Offset 40h, 48h, 50h, 58h, 60h, 68h, 70h, 78h

31	16	15	11	10	8	7	2	1	0
DRAMBase <i>i</i>						reserved	IntlvEn	reserved	WE RE

Bits	Mnemonic	Function	R/W	Reset
31–16	DRAMBase <i>i</i>	DRAM Base Address <i>i</i> Bits 39–24	R/W	X
15–11	reserved		R	0
10–8	IntlvEn	Interleave Enable	R/W	X
7–2	reserved		R	0
1	WE	Write Enable	R/W	0
0	RE	Read Enable	R/W	0

“X” in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Read Enable (RE)**—Bit 0. This bit enables reads to the defined address space.

0 = Disabled  
 1 = Enabled

**Write Enable (WE)**—Bit 1. This bit enables writes to the defined address space.

0 = Disabled  
 1 = Enabled

**Interleave Enable (IntlvEn)**—Bits 10–8. This field enables interleaving on a 4-Kbyte boundary between memory on different nodes. The bits are encoded as follows:

000b = No interleave  
 001b = Interleave on A[12] (2 nodes)  
 010b = reserved  
 011b = Interleave on A[12] and A[13] (4 nodes)  
 100b = reserved  
 101b = reserved  
 110b = reserved  
 111b = Interleave on A[12] and A[13] and A[14] (8 nodes)

**DRAM Base Address *i* Bits 39–24 (DRAMBase*i*)**—Bits 31–16. This field defines the upper address bits of a 40-bit address that defines the start of DRAM region *i* (where *i* = 0, 1, . . . 7).

### 3.4.4.2 DRAM Limit *i* Registers

#### DRAM Limit 0–7 Registers Function 1: Offset 44h, 4Ch, 54h, 5Ch, 64h, 6Ch, 74h, 7Ch

31		16	15		11	10		8	7		3	2	0
DRAMLimit <i>i</i>										reserved	IntlvSel	reserved	DstNode

Bits	Mnemonic	Function	R/W	Reset
31–16	DRAMLimit <i>i</i>	DRAM Limit Address <i>i</i> (39–24)	R/W	X
15–11	reserved		R	0
10–8	IntlvSel	Interleave Select	R/W	X
7–3	reserved		R	0
2–0	DstNode	Destination Node ID	R/W	X

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

#### Field Descriptions

**Destination Node ID (DstNode)**—Bits 2–0. This field specifies the node that a packet is sent to if it is within the address range. The value of this field must correspond to the number of the register pair, and it should be set as follows:

000b for Function 1, Offset 44h register

001b for Function 1, Offset 4Ch register

...

111b for Function 1, Offset 7Ch register

**Interleave Select (IntlvSel)**—Bits 10–8. This field specifies the values of address bits A[14:12] to use with the Interleave Enable field (IntlvEn[2:0]) to determine which 4-Kbyte blocks are routed to this region (See Figure 1).

IntlvSel[0] corresponds to A[12]

IntlvSel[1] corresponds to A[13]

IntlvSel[2] corresponds to A[14]

**DRAM Limit Address *i* (39–24) (DRAMLimit*i*)**—Bits 31–16. This field defines the upper address bits of a 40-bit address that defines the end of DRAM region *n* (where *i* = 0, 1, . . . 7).



<b>Node 0</b> IntlvEn[2:0] = 011 IntlvSel[2:0] = 000 A[13:12] = 00	<b>Node 1</b> IntlvEn[2:0] = 011 IntlvSel[2:0] = 001 A[13:12] = 01
<b>Node 2</b> IntlvEn[2:0] = 011 IntlvSel[2:0] = 010 A[13:12] = 10	<b>Node 3</b> IntlvEn[2:0] = 011 IntlvSel[2:0] = 011 A[13:12] = 11

**Figure 1. Interleave Example (IntlvEn Relation to IntlvSel)**

### 3.4.5 Memory-Mapped I/O Address Map Registers

These registers define sections of the memory address map for which accesses should be routed to memory-mapped I/O. MMIO regions must not overlap each other. For addresses within the specified range of a base/limit pair, requests are routed to the noncoherent HyperTransport link specified by the destination Node ID and destination Link ID.

Addresses are considered to be within the defined range if they are greater than or equal to the base and less than or equal to the limit. For the purposes of this comparison, the lower unspecified bits of the base are assumed to be 0s and the lower unspecified bits of the limit are assumed to be 1s.

An address that maps to both DRAM and memory-mapped I/O is routed to MMIO.

Programming of the MMIO address maps must be consistent with the Top Of Memory and Memory Type Range registers (see Chapter 13, “Processor Configuration Registers”). In particular, accesses from the CPU can only hit in the MMIO address maps if the corresponding CPU memory type is of type IO. For accesses from I/O devices, the lookup is based on address only.

#### 3.4.5.1 Extended Configuration Space Access

Chipset devices may support PCI-defined extended configuration space through an MMIO range. Typically, requests to the MMIO range for extended configuration space are required to use the non-posted channel. Therefore, the Non-Posted bit should be set for this MMIO range. Instructions used to read extended configuration space must be of the following form:

```
mov EAX/AX/AL, <any_address_mode>;
```

Instructions used to write extended configuration space must be of the following form:

```
mov <any_address_mode>, EAX/AX/AL;
```

In addition, all such accesses are required not to cross any naturally aligned doubleword boundary. Access to extended configuration registers that do not meet these requirements result in undefined behavior. Extended configuration space is normally specified to be the uncacheable memory type.

### 3.4.5.2 Memory-Mapped I/O Base *i* Registers

#### Memory-Mapped I/O Base 0–7 Registers

Function 1: Offset 80h, 88h, 90h, 98h,  
A0h, A8h, B0h, B8h

31		8	7	4	3	2	1	0
MMIOBase <i>i</i>				reserved	Lock	CpuDis	WE	RE

Bits	Mnemonic	Function	R/W	Reset
31–8	MMIOBase <i>i</i>	Memory-Mapped I/O Base Address <i>i</i> (39–16)	R/W	X
7–4	reserved		R	0
3	Lock	Lock	R/W	X
2	CpuDis	CPU Disable	R/W	X
1	WE	Write Enable	R/W	0
0	RE	Read Enable	R/W	0

“X” in the Reset column indicates that the field initializes to an undefined state after reset.

#### Field Descriptions

**Read Enable (RE)**—Bit 0. This bit enables reads to the defined address space.

0 = Disabled  
1 = Enabled

**Write Enable (WE)**—Bit 1. This bit enables writes to the defined address space.

0 = Disabled  
1 = Enabled

**CPU Disable (CpuDis)**—Bit 2. When set, this bit causes this MMIO range to apply to I/O requests only, not to CPU requests.

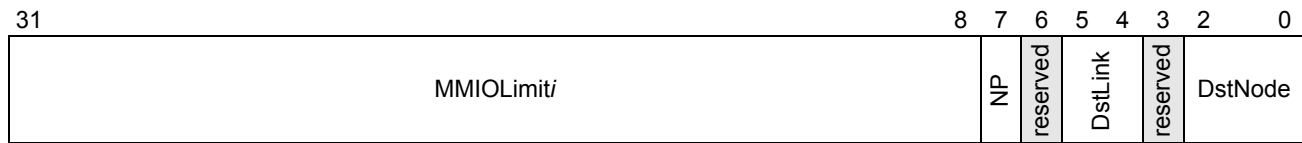
**Lock (Lock)**—Bit 3. Setting this bit, along with either the WE or RE bits, makes the base/limit registers read-only.

**Memory-Mapped I/O Base Address *i* (39–16) (MMIOBase*i*)**—Bit 31–8. This field defines the upper address bits of a 40-bit address that defines the start of memory-mapped I/O region *n* (where *n* = 0, 1, . . . 7).

### 3.4.5.3 Memory-Mapped I/O Limit *i* Registers

#### Memory-Mapped I/O Limit *i* Registers

Function 1: Offset 84h, 8Ch, 94h, 9Ch,  
A4h, ACh, B4h, BCh



Bits	Mnemonic	Function	R/W	Reset
31–8	MMIOLimit <i>i</i>	Memory-Mapped I/O Limit Address <i>i</i>	R/W	X
7	NP	Non-Posted	R/W	X
6	reserved		R	0
5–4	DstLink	Destination Link ID	R/W	X
3	reserved		R	0
2–0	DstNode	Destination Node ID	R/W	X

“X” in the Reset column indicates that the field initializes to an undefined state after reset.

#### Field Descriptions

**Destination Node ID (DstNode)**—Bits 2–0. This field specifies the node that a packet is sent to if it is within the address range. The bits are encoded as follows:

000b = Node 0  
 001b = Node 1  
 ...  
 111b = Node 7

**Destination Link ID (DstLink)**—Bits 5–4. This field specifies the HyperTransport link number to send the packet to once the packet reaches the destination node.

00b = Link 0  
 01b = Link 1  
 10b = Link 2  
 11b = reserved

**Non-Posted (NP)**—Bit 7. When set to 1, this bit forces CPU writes to this memory-mapped I/O region to be non-posted. This may be used to force writes to be non-posted for MMIO regions which map to the legacy ISA/LPC bus, or in conjunction with the DsNpReqLmt field in the HyperTransport Transaction Control register (page 48) in order to allow downstream CPU requests to be counted and thereby limited to a specified number. This latter use of the NP bit may be used to avoid loop deadlock in systems that implement a reflection region in an I/O device that reflects downstream accesses back upstream. See the *HyperTransport™ I/O Link Specification* summary of deadlock scenarios for more information.

0 = CPU writes may be posted  
 1 = CPU writes must be non-posted

**Memory-Mapped I/O Limit Address  $i$  (39–16) (MMIOLimit $i$ )**—Bits 31–8. This field defines the upper address bits of a 40-bit address that defines the end of memory-mapped I/O region  $n$  (where  $n = 0, 1, \dots, 7$ ).

### 3.4.6 PCI I/O Address Map Registers

These registers define sections of the PCI I/O address map for which accesses should be routed to each noncoherent HyperTransport technology chain. PCI I/O regions must not overlap each other. For PCI I/O addresses within the specified range of a base/limit pair, requests are routed to the noncoherent HyperTransport link specified by the destination Node ID and destination Link ID.

Addresses are considered to be within the defined range if they are greater than or equal to the base and less than or equal to the limit. For the purposes of this comparison, the lower unspecified bits of the base are assumed to be 0s and the lower unspecified bits of the limit are assumed to be 1s.

PCI I/O accesses are generated from x86 IN/OUT instructions.

#### 3.4.6.1 PCI I/O Base $i$ Registers

##### PCI I/O Base 0–3 Registers

##### Function 1: Offset C0h, C8h, D0h, D8h

31	25	24	12	11	6	5	4	3	2	1	0		
reserved			PCIIOBase <i>i</i>			reserved			IE	VE	reserved	WE	RE

Bits	Mnemonic	Function	R/W	Reset
31–25	reserved		R	0
24–12	PCIIOBase $i$	PCI I/O Base Address $i$	R/W	X
11–6	reserved		R	0
5	IE	ISA Enable	R/W	X
4	VE	VGA Enable	R/W	X
3–2	reserved		R	0
1	WE	Write Enable	R/W	0
0	RE	Read Enable	R/W	0

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Read Enable (RE)**—Bit 0. This bit enables reads to the defined address space.

0 = Disabled  
1 = Enabled

**Write Enable (WE)**—Bit 1. This bit enables writes to the defined address space.

0 = Disabled

1 = Enabled

**VGA Enable (VE)**—Bit 4. Forces addresses in the first 64 Kbytes of PCI I/O space and where A[9:0] is in the range 3B0–3BBh or 3C0–3DFh to match against this base/limit pair independent of the base/limit addresses (see the *PCI-to-PCI Bridge Architecture* specification for a description of this function). A WE or RE bit must also be set to enable this feature.

The MMIO 000A\_0000h to 000B\_FFFFh matching function normally associated with the VGA enable bit in PCI is NOT included in the VE bit. To enable this behavior, an MMIO register pair must be used.

**ISA Enable (IE)**—Bit 5. Blocks addresses in the first 64 Kbytes of PCI I/O space and in the last 768 bytes in each 1-Kbyte block from matching against this base/limit pair (see the *PCI-to-PCI Bridge Architecture* specification for a description of this function). A WE or RE bit must also be set to enable this feature.

**PCI I/O Base Address *i* (PCIIOBase*i*)**—Bits 24–12. This field defines the start of PCI I/O region *n* (where *n* = 0, 1, 2, 3).

### 3.4.6.2 PCI I/O Limit *i* Registers

#### PCI I/O Limit 0–3 Registers

#### Function 1: Offset C4h, CCh, D4h, DCh

31	25	24	12	11	6	5	4	3	2	0
reserved					reserved			DstLink	reserved	DstNode

Bits	Mnemonic	Function	R/W	Reset
31–25	reserved		R	0
24–12	PCIIOLimit <i>i</i>	PCI I/O Limit Address <i>i</i>	R/W	X
11–6	reserved		R	0
5–4	DstLink	Destination Link ID	R/W	X
3	reserved		R	0
2–0	DstNode	Destination Node ID	R/W	X

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

### Field Descriptions

**Destination Node ID (DstNode)**—Bits 2–0. This field specifies the node to which a packet is sent if it is within the address range. The bits are encoded as follows:

000b = Node 0  
 001b = Node 1  
 ...  
 111b = Node 7

**Destination Link ID (DstLink)**—Bits 5–4. This field specifies the link to send the packet to once the packet has reached the destination node.

00b = Link 0  
01b = Link 1  
10b = Link 2  
11b = reserved

**PCI I/O Limit Address *i* (PCI I/OLimit<sub>*i*</sub>)**—Bits 24–12. This field defines the end of PCI I/O region

### 3.4.7 Configuration Map Registers

These registers define sections of the configuration bus number map for which configuration accesses should be routed to each noncoherent HyperTransport technology chain. Configuration regions must not overlap each other. For configuration accesses with bus numbers within the specified range of a base/limit pair, requests are routed to the noncoherent HyperTransport link specified by the destination Node ID and destination Link ID.

Bus numbers are considered to be within the defined range if they are greater than or equal to the base and less than or equal to the limit.

In device number compare mode (see DevCmpEn bit), the base/limit fields are interpreted as device number base/limit values for Bus 0 configuration accesses. This may be used to support configurations where Bus 0 is split between two independent noncoherent HyperTransport technology chains.

Configuration accesses to Bus 0, device 24–31, are assumed to be targeting coherent HyperTransport technology devices and are routed to the corresponding coherent HyperTransport node irrespective of the values in these registers (see Chapter 3, “Memory System Configuration”). Note that the range of device numbers affected may be modified based on the number of nodes present (see “HyperTransport™ Transaction Control Register” on page 48).

#### Configuration Base and Limit 0–3 Registers      Function 1: Offset E0h, E4h, E8h, ECh

31	24	23	16	15	10	9	8	7	6	4	3	2	1	0		
BusNumLimit <i>i</i>				BusNumBase <i>i</i>				reserved		DstLink	reserved	DstNode	reserved	DevCmpEn	WE	RE

Bits	Mnemonic	Function	R/W	Reset
31–24	BusNumLimit <sub><i>i</i></sub>	Bus Number Limit <i>i</i>	R/W	X
23–16	BusNumBase <sub><i>i</i></sub>	Bus Number Base <i>i</i>	R/W	X
15–10	reserved		R	0
9–8	DstLink	Destination Link ID	R/W	X
7	reserved		R	0
6–4	DstNode	Destination Node ID	R/W	X

Bits	Mnemonic	Function	R/W	Reset
3	reserved		R	0
2	DevCmpEn	Device Number Compare Enable	R/W	X
1	WE	Write Enable	R/W	0
0	RE	Read Enable	R/W	0

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

## Field Descriptions

**Read Enable (RE)**—Bit 0. This bit enables reads to the defined address space.

0 = Disabled

1 = Enabled

**Write Enable (WE)**—Bit 1. This bit enables writes to the defined address space.

0 = Disabled

1 = Enabled

**Device Number Compare Enable (DevCmpEn)**—Bit 2. When this bit is set, this register defines a device number range rather than a bus number range. To match, configuration cycles must be to bus number 0 and have device numbers between BusNumBase and BusNumLimit. This is used to enable multiple noncoherent HyperTransport chains to be configured as Bus 0.

**Destination Node ID (DstNode)**—Bits 6–4. This field specifies the node that a packet is sent to if it is within the address range. The bits are encoded as follows:

000b = Node 0

001b = Node 1

...

111b = Node 7

**Destination Link ID (DstLink)**—Bits 9–8. This field specifies the link to send the packet to once it has reached the destination node and unit.

00b = Link 0

01b = Link 1

10b = Link 2

11b = reserved

**Bus Number Base *i* (BusNumBase<sub>*i*</sub>)**—Bits 23–16. This field defines the lowest bus number in configuration region *i* (where *i* = 0, 1, 2, 3).

**Bus Number Limit *i* (BusNumLimit<sub>*i*</sub>)**—Bits 31–24. This bit field defines the highest bus number in configuration region *i* (where *i* = 0, 1, 2, 3).

### 3.4.8 DRAM Hole Address Register

This register is used to control memory hoisting. See Section 3.5.8 on page 99 for more information on memory hoisting. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this register.

#### DRAM Hole Address Register

#### Function 1: Offset F0h

31	24	23	16	15	8	7	1	0
DramHoleBase				reserved				DramHoleValid

Bits	Mnemonic	Function	R/W	Reset
31–24	DramHoleBase	DRAM Hole Base Address	R/W	0
23–16	reserved		R	0
15–8	DramHoleOffset	DRAM Hole Offset Address	R/W	0
7–1	reserved		R	0
0	DramHoleValid	DRAM Hole Valid	R/W	0

#### Field Descriptions

**DRAM Hole Valid (DramHoleValid)**—Bit 0. This bit should be set in the processor node(s) that own the DRAM address space that is hoisted above the 4GB address level. If node interleaving is enabled, then this should be set in all processors. This bit should not be set until MemClrStatus (Function 2: Offset 90 bit 11) has been set by hardware.

0 = Memory hoisting is not enabled.

1 = Memory hoisting is enabled in the processor node.

**DRAM Hole Offset (DramHoleOffset)**—Bits 15–8. When memory hosting is enabled, this value is subtracted from the physical address of certain transactions before being passed to the DRAM controller.

**DRAM Hole Base (DramHoleBase)**—Bit 31–24. This specifies the base address of the IO hole, below the 4G address level, that is used in memory hoisting. Normally,  
DramHoleBase >= TOP\_MEM[31:24].

## 3.5 Function 2—DRAM Controller

Function 2 configuration registers are listed in Table 4.



**Table 4. Function 2 Configuration Registers**

Offset	Register Name				Reset	Access	Description
00h	Device ID		Vendor ID (AMD)		1102_1022h	RO	page 82
04h	Status <sup>1</sup>		Command		0000_0000h	RO	
08h	Base Class Code	Subclass Code	Program Interface	Revision ID	0600_0000h	RO	page 83
0Ch	BIST	Header Type	Latency Timer	Cache Line Size	0080_0000h	RO	page 83
10h	Base Address 0				0000_0000h	RO	
14h	Base Address 1				0000_0000h	RO	
18h	Base Address 2				0000_0000h	RO	
1Ch	Base Address 3				0000_0000h	RO	
20h	Base Address 4				0000_0000h	RO	
24h	Base Address 5				0000_0000h	RO	
28h	Card Bus CIS Pointer				0000_0000h	RO	
2Ch	Sub-System ID		Sub-System Vendor ID		0000_0000h	RO	
30h	ROM Base Address				0000_0000h	RO	
34h	Capabilities				0000_0000h	RO	
38h	reserved				0000_0000h	RO	
3Ch	Max Latency	Min GNT	Int Pin	Int Line	0000_0000h	RO	
40h	DRAM CS Base 0				0000_0000h	RW	page 84
44h	DRAM CS Base 1				0000_0000h	RW	page 84
48h	DRAM CS Base 2				0000_0000h	RW	page 84
4Ch	DRAM CS Base 3				0000_0000h	RW	page 84
50h	DRAM CS Base 4				0000_0000h	RW	page 84
54h	DRAM CS Base 5				0000_0000h	RW	page 84
58h	DRAM CS Base 6				0000_0000h	RW	page 84
5Ch	DRAM CS Base 7				0000_0000h	RW	page 84
60h	DRAM CS Mask 0				0000_0000h	RW	page 87
64h	DRAM CS Mask 1				0000_0000h	RW	page 87
68h	DRAM CS Mask 2				0000_0000h	RW	page 87
6Ch	DRAM CS Mask 3				0000_0000h	RW	page 87

**Notes:**

1. The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.
2. Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.

**Table 4. Function 2 Configuration Registers (Continued)**

Offset	Register Name	Reset	Access	Description
70h	DRAM CS Mask 4	0000_0000h	RW	page 87
74h	DRAM CS Mask 5	0000_0000h	RW	page 87
78h	DRAM CS Mask 6	0000_0000h	RW	page 87
7Ch	DRAM CS Mask 7	0000_0000h	RW	page 87
80h	DRAM Bank Address Mapping	0000_0000h	RW	page 88
88h	DRAM Timing Low	0000_0000h	RW	page 102
8Ch	DRAM Timing High	0000_0000h	RW	page 104
90h	DRAM Configuration Low	0000_0000h	RW	page 106
94h	DRAM Configuration High	0000_0000h	RW	page 112
98h	DRAM Delay Line	0000_0000h	RW	page 115
9Ch	Scratch Register	0000_0000h	RW	page 117
<b>Notes:</b> <ol style="list-style-type: none"> <li>1. The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.</li> <li>2. Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.</li> </ol>				

### 3.5.1 Device/Vendor ID Register

This register specifies the device and vendor IDs for the Function 2 registers and is part of the standard PCI configuration header.

#### Device/Vendor ID Register

**Function 2: Offset 00h**

31	16	15	0
DevID		VenID	

Bits	Mnemonic	Function	R/W	Reset
31–16	DevID	Device ID	R	1102h
15–0	VenID	Vendor ID	R	1022h

#### Field Descriptions

**Vendor ID (VenID)**—Bits 15–0. This read-only value is defined as 1022h for AMD.

**Device ID (DevID)**—Bits 31–16. This read-only value is defined as 1102h for the HyperTransport technology configuration function.

### 3.5.2 Class Code/Revision ID Register

This register specifies the class code and revision for the Function 2 registers and is part of the standard PCI configuration header.

#### Class Code/Revision ID Register

Function 2: Offset 08h

31	24	23	16	15	8	7	0	
BCC				SCC		PI		RevID

Bits	Mnemonic	Function	R/W	Reset
31–24	BCC	Base Class Code	R	06h
23–16	SCC	Subclass Code	R	00h
15–8	PI	Programming Interface	R	00h
7–0	RevID	Revision ID	R	00h

#### Field Descriptions

**Revision ID (RevID)**—Bits 7–0.

**Programming Interface (PI)**—Bits 15–8. This read-only value is defined as 00h.

**Sub Class Code (SCC)**—Bits 23–16. This read-only value is defined as 00h.

**Base Class Code (BCC)**—Bits 31–24. This read-only value is defined as 06h for a host bridge device.

### 3.5.3 Header Type Register

This register specifies the header type for the Function 2 registers and is part of the standard PCI configuration header.

#### Header Type Register

Function 2: Offset 0Ch

31	24	23	16	15	8	7	0	
BIST				HType		LatTimer		CLS

Bits	Mnemonic	Function	R/W	Reset
31–24	BIST	BIST	R	00h
23–16	HType	Header Type	R	80h
15–8	LatTimer	Latency Timer	R	00h
7–0	CLS	Cache Line Size	R	00h

## Field Descriptions

**CacheLineSize (CLS)**—Bits 7–0. This read-only value is defined as 00h.

**LatencyTimer (LatTimer)**—Bits 15–8. This read-only value is defined as 00h.

**HeaderType (Htype)**—Bits 23–16. This read-only value is defined as 80h to indicate that multiple functions are present in the configuration header and that the header layout corresponds to a device header as opposed to a bridge header. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**BIST**—Bits 31–24. This read-only value is defined as 00h.

### 3.5.4 DRAM CS Base Address Registers

These registers define DRAM chip select (CS) address mapping. They are programmed based on data in the Serial Presence Detect (SPD) ROM on each DIMM.

Function 1 address map registers define to which node to route a DRAM request. Function 2 address map registers define what DRAM chip select to access on that particular node. See “DRAM Address Map” on page 70. for more information on addresses routed to the DRAM controller (InputAddr).

Memory size of each chip select bank is defined by a DRAM CS Base Address Register and a DRAM CS Mask register. The chip selects are formed as follows, using the field names from the DRAM CS Base Address Registers and DRAM CS Mask Registers, where

“,” means “concatenate”

“/” means “not”

“==” means “equals”

“&” means “and”

ChipSelect[i] is asserted if the following is true:

```
CSBE[i] &
( { InputAddr[35:34], (InputAddr[33:25] & /AddrMaskHi[i][33:25]),
  ( InputAddr[19:13] & /AddrMaskLo[i][19:13]) } ==
  { BaseAddrHi[i][35:34], (BaseAddrHi[i][33:25] & /AddrMaskHi[i][33:25]),
    ( BaseAddrLo[i][19:13] & /AddrMaskLo[i][19:13]) } ) );
```

There are eight DRAM CS Base Address and DRAM CS Mask register pairs. DRAM CS Base Address 0 and DRAM CS Mask 0 define the base address for chip select 0, and so on. DIMM0 is associated with CS0 and CS1, DIMM1 is associated with CS2 and CS3, and so on. If only DIMM2 and DIMM3 are occupied and DIMM2 is connected to CS[5:4] and DIMM3 is connected to CS[7:6], then bit 0 of DRAM CS Base Address Register[7-4] is set to 1 to enable these chip-select banks.

**Table 5. DRAM CS Base Address and DRAM CS Mask Registers**

Registers	Chip Selects	Memory Module D[127:64] <sup>1</sup>	Memory Module D[63:0]
DRAM CS Base and Mask 0	CS0	Extended DIMM0	DIMM0
DRAM CS Base and Mask 1	CS1	Extended DIMM0	DIMM0
DRAM CS Base and Mask 2	CS2	Extended DIMM1	DIMM1
DRAM CS Base and Mask 3	CS3	Extended DIMM1	DIMM1
DRAM CS Base and Mask 4	CS4	Extended DIMM2/DIMM0 <sup>2</sup>	DIMM2/DIMM0 <sup>2</sup>
DRAM CS Base and Mask 5	CS5	Extended DIMM2/DIMM0 <sup>2</sup>	DIMM2/DIMM0 <sup>2</sup>
DRAM CS Base and Mask 6	CS6	Extended DIMM3/DIMM1 <sup>2</sup>	DIMM3/DIMM1 <sup>2</sup>
DRAM CS Base and Mask 7	CS7	Extended DIMM3/DIMM1 <sup>2</sup>	DIMM3/DIMM1 <sup>2</sup>
<b>Notes:</b> <ol style="list-style-type: none"> <li>1. This column only applies to processors configured for a 128-bit DRAM interface.</li> <li>2. Quad Rank RDIMM.</li> </ol>			

**Table 6. DRAM CS Base Address and DRAM CS Mask Registers Mismatched DIMM Support<sup>1</sup>**

Registers	Chip Selects	Memory Module D[127:64]	Memory Module D[63:0]
DRAM CS Base and Mask 0	CS0	N/A	DIMM0
DRAM CS Base and Mask 1	CS1	N/A	DIMM0
DRAM CS Base and Mask 2	CS2	N/A	DIMM1
DRAM CS Base and Mask 3	CS3	N/A	DIMM1
DRAM CS Base and Mask 4	CS4	DIMM2	N/A
DRAM CS Base and Mask 5	CS5	DIMM2	N/A
DRAM CS Base and Mask 6	CS6	DIMM3	N/A
DRAM CS Base and Mask 7	CS7	DIMM3	N/A
<b>Notes:</b> <ol style="list-style-type: none"> <li>1. Mod64BitMux (Function 2, Offset 90 Bit 6). must be set to one to support this mode.</li> </ol>			

DRAM memory can be assigned to chip select banks in two ways: non-interleaving, when contiguous addresses are assigned to each chip select bank, and interleaving, when non-contiguous addresses are assigned to each chip select bank.

Non-interleaving mode can always be used. The BIOS must assign the largest DIMM chip-select range to the lowest address. As addresses increase, the chip select size must remain constant or decrease. This is necessary to keep DIMM chip select banks on aligned address boundaries as chip-select banks with different depths are added. The masking does not work otherwise.

Memory interleaving mode can be used if the memory system is composed of a single type and size DRAM, and if the number of chip selects is a power of two. DRAM controller swaps low order address bits with high order address bits during decoding. As a result, some low order address bits are used to determine chip select, and some high order address bits are used to determine DIMM row. It allows that a switch between DIMMs occurs before a page conflict within a DIMM. The one cycle turnaround saves time needed to close/open new pages. Algorithm for programming DRAM CS Base Address and DRAM CS Mask registers in interleaving mode are described in section “DRAM Address Mapping in Interleaving Mode” on page 95.

## DRAM CS Base Address Registers

**Function 2: Offset 40h, 44h, 48h, 4Ch,  
50h, 54h, 58h, 5Ch**

31	21	20	16	15	9	8	1	0	
BaseAddrHi				reserved	BaseAddrLo			reserved	CSBE

Bits	Mnemonic	Function	R/W	Reset
31–21	BaseAddrHi	Base Address (35–25)	R/W	0
20–16	reserved		R	0
15–9	BaseAddrLo	Base Address (19–13)	R/W	0
8–1	reserved		R	0
0	CSBE	Chip-Select Bank Enable	R/W	0

## Field Descriptions

**Chip-Select Bank Enable (CSBE)**—Bit 0. This bit enables access to the defined address space.

**Base Address (19–13) (BaseAddrLo)**—Bits 15–9. This field is only used for memory interleaving to avoid page conflicts. A new chip select bank is accessed before accessing a new row in the old chip select bank, delaying or avoiding a page conflict.

If contiguous addresses from address 0 are accessed, the controller would first access a row (e.g., row 0) in bank 0, chip select 0. As the address increases, the controller would access different columns in the following order:

row 0/bank 0/chip select 0 and then  
row 0/bank 1/chip select 0 and then  
row 0/bank 2/chip select 0 and then  
row 0/bank 3/chip select 0 and then  
row 0/bank 0/chip select 1.

A chip select bank does not have a contiguous region of memory assigned to it. Memory is interleaved between two DIMMs, four DIMMs, or eight DIMMs.

**Base Address (35–25) (BaseAddrHi)**—Bits 31–21. These bits decode 32-Mbyte blocks of memory. In the non-interleaving mode (BaseAddrLo has a value of 0), physical addresses map to DRAM addresses in the following way:

Col—Lowest order physical address bits

Bank—Second lowest order physical address bits (page miss is better than page conflict)

Row—Second highest order physical address bits (page conflict before accessing new chip)

Chip Select—Highest order physical address bits

In the non-interleaving mode, contiguous addresses from 0 would first access a row (e.g., row 0) in bank 0, chip select 0. As the address increases, the next access is to a different column and eventually a different chip select bank, e.g.:

row 0/bank 0/chip select 0 and then

row 0/bank 1/chip select 0 and then

row 0/bank 2/chip select 0 and then

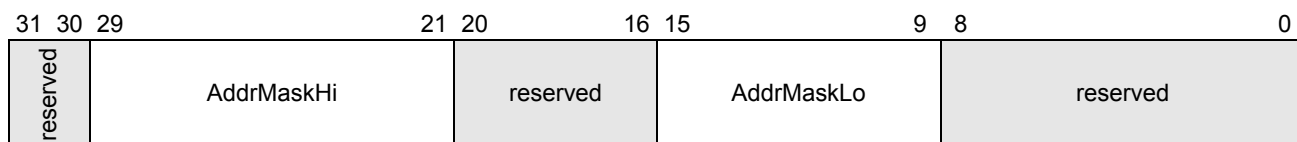
row 0/bank 3/chip select 0 and then

row 1/bank 0/chip select 0

Memory controller accesses all rows and banks in a single chip select bank before accessing a new chip select bank in the non-interleaving mode.

### 3.5.5 DRAM CS Mask Registers

The purpose of this register is to exclude the corresponding address bits from the comparison with the DRAM Base address register.

**DRAM CS Mask Registers      Function 2: Offset 60h, 64h, 68h, 6Ch, 70h, 74h, 78h, 7Ch**

Bits	Mnemonic	Function	R/W	Reset
31–30	reserved		R	0
29–21	AddrMaskHi	Address Mask (33–25)	R/W	0
20–16	reserved		R	0
15–9	AddrMaskLo	Address Mask (19–13)	R/W	0
8–0	reserved		R	0

## Field Descriptions

**Address Mask (19–13) (AddrMaskLo)**—Bits 15–9. This field specifies the addresses to be excluded for the memory interleaving mode described in the Base Address bit definitions.

**Address Mask (33–25) (AddrMaskHi)**—Bits 29–21. This field defines the top Address Mask bits.

The bits with an address mask of 1 are excluded from the address comparison. This allows the memory block size to be larger than 32 Mbytes. If Address Mask bit 25 is set to 1, the memory block size is 64 Mbytes.

### 3.5.6 DRAM Bank Address Mapping Register

The DIMM may have one or two chip-select banks, but it always has support for up to two chip-select banks. The address mode covers the whole DIMM, regardless of whether there is one or two chip-select banks.

#### DRAM CS Address Mapping Register

**Function 2: Offset 80h**

31	30	29		16	15		12	11		8	7		4	3	0
reserved	BankSwizzleMode	reserved													
								CS7/6							CS1/0

Bits	Mnemonic	Function	R/W	Reset
31	reserved		R	0
30	BankSwizzleMode	Bank Swizzle Mode	R/W	0
29–16	reserved		R	0
15–12	CS7/6	CS7/6	R/W	0
11–8	CS5/4	CS5/4	R/W	0
7–4	CS3/2	CS3/2	R/W	0
3–0	CS1/0	CS1/0	R/W	0

#### Field Descriptions

**Chip Select 1/0 (CS1/0)**—Bits 3–0. This field specifies the memory module size. This field is programmed the same regardless of whether the DRAM interface is 64-bits or 128-bits wide. This field describes the type of DIMMs that compose the chip select.

Revision CG and earlier revisions Chip Select Bank Address Mode Encoding:

000b = 32 Mbyte (Rows =12 & Col=8)

001b = 64 Mbyte (Rows =12 & Col=9)

010b = 128 Mbyte (Rows =13 & Col=9) | (Rows=12 & Col=10)

011b = 256 Mbyte (Rows =13 & Col=10) | (Rows=12 & Col=11)

100b = 512 Mbyte (Rows =13 & Col=11) | (Rows=14 & Col=10)

101b = 1 Gbyte (Rows =14 & Col=11) | (Rows=13 & Col=12)



110b = 2 Gbyte (Rows =14 & Col=12)

111b = reserved

Revision D and later revisions Chip Select Bank Address Mode Encoding:

0000b = 32 Mbyte (Rows =12 & Col=8)

0001b = 64 Mbyte (Rows =12 & Col=9)

0010b = 128 Mbyte (Rows =13 & Col=9)

0011b = 128 Mbyte (Rows =12 & Col=10)

0100b = 256 Mbyte (Rows =13 & Col=10)

0101b = 512 Mbyte (Rows =14 & Col=10)

0110b = 256 Mbyte (Rows =12 & Col=11)

0111b = 512 Mbyte (Rows =13 & Col=11)

1000b = 1 Gbyte (Rows =14 & Col=11)

1001b = 1 Gbyte (Rows =13 & Col=12)

1010b = 2 Gbyte (Rows =14 & Col=12)

**Chip Select 3/2 (CS3/2)**—Bits 7–4. This field specifies the memory module size. The bits are encoded as shown in CS1/0.

**Chip Select 5/4 (CS5/4)**—Bits 11–8. This field specifies the memory module size. The bits are encoded as shown in CS1/0.

**Chip Select 7/6 (CS7/6)**—Bits 15–12. This field specifies the memory module size. The bits are encoded as shown in CS1/0.

**Bank Swizzle Mode (BankSwizzleMode)**—Bit 30. This bit enables a DRAM bank address mode that XORs address bits to create the bank address. This mode reduces page conflicts between a cache line fill and a cache line evict. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this register.

### 3.5.6.1 DRAM Address Mapping in Non-Interleaving Mode

The address line mapping for non-interleaving mode with a 64-bit DRAM interface is shown in Table 7 for revision CG and earlier revisions and in Table 9 for revision D and later revisions. The address line mapping for non-interleaving mode with a 128-bit DRAM interface in Table 8 for revision CG and earlier revisions and in Table 10 for revision D and later revisions. The address line mapping for revision E non-interleaving and bank swizzle mode with a 64-bit DRAM interface is shown in Table 11 and with a 128-bit DRAM interface is shown in Table 12. Column A10 presents the precharge all signal (PC).

**Table 7. Mapping Host Address Lines to Memory Address Lines with 64-Bit Interface (revision CG and earlier revisions)**

**64-Bit Interface to Memory (revision CG and earlier revisions)**

CS Mode	CS Size	Device size, width	Bank		Address														
			1	0		13	12	11	10	9	8	7	6	5	4	3	2	1	0
000b	32MB	64Mb, x16	12	11	Row	X	X	18	17	16	15	14	13	24	23	22	21	20	19
					Col	X	X	X	PC	X	X	10	9	8	7	6	5	4	3
001b	64MB	64Mb, x8 128Mb, x16	12	13	Row	X	X	18	17	16	15	14	25	24	23	22	21	20	19
					Col	X	X	X	PC	X	11	10	9	8	7	6	5	4	3
010b	128MB	64Mb, x4 128Mb, x8 256Mb, x16	12	13	Row	X	26	18	17	16	15	14	25	24	23	22	21	20	19
					Col	X	X	X	PC	26	11	10	9	8	7	6	5	4	3
011b	256MB	128Mb, x4 256Mb, x8 512Mb, x16	14	13	Row	X	27	18	17	16	15	26	25	24	23	22	21	20	19
					Col	X	X	27	PC	12	11	10	9	8	7	6	5	4	3
100b	512MB	256Mb, x4 512Mb, x8 1Gb, x16	14	13	Row	28	27	18	17	16	15	26	25	24	23	22	21	20	19
					Col	X	X	28	PC	12	11	10	9	8	7	6	5	4	3
101b	1GB	512Mb, x4 1Gb, x8	14	15	Row	28	27	18	17	16	29	26	25	24	23	22	21	20	19
					Col	X	28	13	PC	12	11	10	9	8	7	6	5	4	3
110b	2GB	1Gb, x4	14	15	Row	28	27	18	17	16	29	26	25	24	23	22	21	20	19
					Col	X	30	13	PC	12	11	10	9	8	7	6	5	4	3

**Table 8. Mapping Host Address Lines to Memory Address Lines with 128-Bit Interface (revision CG and earlier revisions)**

**128-Bit Interface to Memory (revision CG and earlier revisions)**

CS Mode	CS Size	Device size, width	Bank		Address														
			1	0		13	12	11	10	9	8	7	6	5	4	3	2	1	0
000b	64MB	64Mb, x16	13	12	Row	X	X	19	18	17	16	15	14	25	24	23	22	21	20
					Col	X	X	X	PC	X	X	11	10	9	8	7	6	5	4
001b	128MB	64Mb, x8 128Mb, x16	13	14	Row	X	X	19	18	17	16	15	26	25	24	23	22	21	20
					Col	X	X	X	PC	X	12	11	10	9	8	7	6	5	4

**Table 8. Mapping Host Address Lines to Memory Address Lines with 128-Bit Interface (revision CG and earlier revisions)**

128-Bit Interface to Memory (revision CG and earlier revisions)

CS Mode	CS Size	Device size, width	Bank		Address														
			1	0		13	12	11	10	9	8	7	6	5	4	3	2	1	0
010b	256MB	64Mb, x4 128Mb, x8 256Mb, x16	13	14	Row	X	27	19	18	17	16	15	26	25	24	23	22	21	20
					Col	X	X	X	PC	27	12	11	10	9	8	7	6	5	4
011b	512MB	128Mb, x4 256Mb, x8 512Mb, x16	15	14	Row	X	28	19	18	17	16	27	26	25	24	23	22	21	20
					Col	X	X	28	PC	13	12	11	10	9	8	7	6	5	4
100b	1GB	256Mb, x4 512Mb, x8 1Gb, x16	15	14	Row	29	28	19	18	17	16	27	26	25	24	23	22	21	20
					Col	X	X	29	PC	13	12	11	10	9	8	7	6	5	4
101b	2GB	512Mb, x4 1Gb, x8	15	16	Row	29	28	19	18	17	30	27	26	25	24	23	22	21	20
					Col	X	29	14	PC	13	12	11	10	9	8	7	6	5	4
110b	4GB	1Gb, x4	15	16	Row	29	28	19	18	17	30	27	26	25	24	23	22	21	20
					Col	X	31	14	PC	13	12	11	10	9	8	7	6	5	4

**Table 9. Mapping Host Address Lines to Memory Address Lines with 64-Bit Interface (revision D and later revisions)**

64-Bit Interface to Memory (revision D and later revisions)

CS Mode	CS Size	Device size, width	Bank		Address														
			1	0		13	12	11	10	9	8	7	6	5	4	3	2	1	0
0000b	32MB	64Mb, x16	12	11	Ro w	X	X	18	17	16	15	14	13	24	23	22	21	20	19
					Col	X	X	X	PC	X	X	10	9	8	7	6	5	4	3
0001b	64MB	64Mb, x8	13	12	Ro w	X	26	18	17	16	15	14	25	24	23	22	21	20	19
		128Mb, x16			Col	X	X	X	PC	X	11	10	9	8	7	6	5	4	3
0010b	128MB	256Mb, x16																	
0011b	128MB	64Mb, x4	14	13	Ro w	28	27	18	17	16	15	26	25	24	23	22	21	20	19
		128Mb, x8			Col	X	X	X	PC	12	11	10	9	8	7	6	5	4	3
0100b	256MB	256Mb, x8 512Mb, x16																	
0101b	512MB	1Gb, x16																	

**Table 9. Mapping Host Address Lines to Memory Address Lines with 64-Bit Interface (revision D and later revisions)**
**64-Bit Interface to Memory (revision D and later revisions)**

CS Mode	CS Size	Device size, width	Bank		Address														
			1	0		13	12	11	10	9	8	7	6	5	4	3	2	1	0
0110b	256MB	128Mb, x4	15	14	Row	29	28	18	17	16	27	26	25	24	23	22	21	20	19
0111b	512MB	256Mb, x4			Col	X	X	13	PC	12	11	10	9	8	7	6	5	4	3
		512Mb, x8																	
1000b	1GB	1Gb, x8	16	15															
1001b	1GB	512Mb, x4			Row	30	29	18	17	28	27	26	25	24	23	22	21	20	19
1010b	2GB	1Gb, x4			Col	X	14	13	PC	12	11	10	9	8	7	6	5	4	3

**Table 10. Mapping Host Address Lines to Memory Address Lines with 128-Bit Interface (revision D and later revisions)**
**128-Bit Interface to Memory (revision D and later revisions)**

CS Mode	CS Size	Device size, width	Bank		Address														
			1	0		13	12	11	10	9	8	7	6	5	4	3	2	1	0
0000b	64MB	64Mb, x16	13	12	Ro w	X	X	19	18	17	16	15	14	25	24	23	22	21	20
					Col	X	X	X	PC	X	X	11	10	9	8	7	6	5	4
0001b	128MB	64Mb, x8	14	13	Ro w	X	27	19	18	17	16	15	26	25	24	23	22	21	20
		128Mb, x16			Col	X	X	X	PC	X	12	11	10	9	8	7	6	5	4
0010b	256MB	256Mb, x16																	
0011b	256MB	64Mb, x4	15	14	Ro w	29	28	19	18	17	16	27	26	25	24	23	22	21	20
		128Mb, x8			Col	X	X	X	PC	13	12	11	10	9	8	7	6	5	4
0100b	512MB	256Mb, x8 512Mb, x16																	
0101b	1GB	1Gb, x16																	
0110b	512MB	128Mb, x4	16	15	Ro w	30	29	19	18	17	28	27	26	25	24	23	22	21	20
0111b	1GB	256Mb, x4 512Mb, x8			Col	X	X	14	PC	13	12	11	10	9	8	7	6	5	4
1000b	2GB	1Gb, x8																	
1001b	2GB	512Mb, x4	17	16	Ro w	31	30	19	18	29	28	27	26	25	24	23	22	21	20
1010b	4GB	1Gb, x4			Col	X	15	14	PC	13	12	11	10	9	8	7	6	5	4

**Table 11. Mapping Host Address Lines to Memory Address Lines with 64-Bit Interface Bank Swizzle Mode (Revision E and later revisions)**
**64-Bit Interface to Memory (Revision E and later revisions)**

CS Mode	CS Size	Device size, width	Bank		Address															
			1	0		13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0000b	32MB	64Mb, x16	12 xor 18 xor 21	11 xor 17 xor 20	Row	X	X	18	17	16	15	14	13	24	23	22	21	20	19	
					Col	X	X	X	PC	X	X	10	9	8	7	6	5	4	3	
0001b	64MB	64Mb, x8	13 xor 18 xor 21	12 xor 17 xor 20	Row	X	26	18	17	16	15	14	25	24	23	22	21	20	19	
		Col			X	X	X	PC	X	11	10	9	8	7	6	5	4	3		
0010b	128MB	256Mb, x16																		
0011b	128MB	64Mb, x4	14 xor 18 xor 21	13 xor 17 xor 20	Row	28	27	18	17	16	15	26	25	24	23	22	21	20	19	
		Col			X	X	X	PC	12	11	10	9	8	7	6	5	4	3		
0100b	256MB	256Mb, x8 512Mb, x16																		
0101b	512MB	1Gb, x16																		
0110b	256MB	128Mb, x4	15 xor 18 xor 21	14 xor 17 xor 20	Row	29	28	18	17	16	27	26	25	24	23	22	21	20	19	
0111b	512MB	256Mb, x4 512Mb, x8			Col	X	X	13	PC	12	11	10	9	8	7	6	5	4	3	
1000b	1GB	1Gb, x8																		
1001b	1GB	512Mb, x4	16 xor 18 xor 21	15 xor 17 xor 20	Row	30	29	18	17	28	27	26	25	24	23	22	21	20	19	
1010b	2GB	1Gb, x4			Col	X	14	13	PC	12	11	10	9	8	7	6	5	4	3	

**Table 12. Mapping Host Address Lines to Memory Address Lines with 128-Bit Interface Bank Swizzle Mode (Revision E and later revisions)**
**128-Bit Interface to Memory (Revision E and later revisions)**

CS Mode	CS Size	Device size, width	Bank		Address															
			1	0		13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0000b	64MB	64Mb, x16	13 xor 19 xor 22	12 xor 18 xor 21	Row	X	X	19	18	17	16	15	14	25	24	23	22	21	20	
					Col	X	X	X	PC	X	X	11	10	9	8	7	6	5	4	
0001b	128MB	64Mb, x8	14 xor 19 xor 22	13 xor 18 xor 21	Row	X	27	19	18	17	16	15	26	25	24	23	22	21	20	
		Col			X	X	X	PC	X	12	11	10	9	8	7	6	5	4		
0010b	256MB	256Mb, x16																		

**Table 12. Mapping Host Address Lines to Memory Address Lines with 128-Bit Interface Bank Swizzle Mode (Revision E and later revisions)**
**128-Bit Interface to Memory (Revision E and later revisions)**

CS Mode	CS Size	Device size, width	Bank		Address															
			1	0		13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0011b	256MB	64Mb, x4	15 xor 19 xor 22	14 xor 18 xor 21	Row	29	28	19	18	17	16	27	26	25	24	23	22	21	20	
		Col			X	X	X	PC	13	12	11	10	9	8	7	6	5	4		
0100b	512MB	256Mb, x8 512Mb, x16																		
0101b	1GB	1Gb, x16																		
0110b	512MB	128Mb, x4	16 xor 19 xor 22	15 xor 18 xor 21	Row	30	29	19	18	17	28	27	26	25	24	23	22	21	20	
0111b	1GB	256Mb, x4 512Mb, x8			Col	X	X	14	PC	13	12	11	10	9	8	7	6	5	4	
1000b	2GB	1Gb, x8																		
1001b	2GB	512Mb, x4	17 xor 19 xor 22	16 xor 18 xor 21	Row	31	30	19	18	29	28	27	26	25	24	23	22	21	20	
1010b	4GB	1Gb, x4			Col	X	15	14	PC	13	12	11	10	9	8	7	6	5	4	

The BIOS sets up the base address and base mask registers after determining the number of populated DIMMs, how many chip-select banks they have, and their sizes. A single base address and mask are initialized for each chip select. Hardware decodes addresses and determines to which DIMM chip-select range the address maps, as a function of the base address and mask.

**Example.** DRAM memory consists of:

- DIMM0, 2-sided with 128 MBytes on each side (total DIMM0 memory is 256 Mbytes)
- DIMM1, 2-sided with 256 MBytes on each side (total DIMM1 memory: 512 Mbyte)
- DIMM2, 2-sided with 64 MBytes on each side (total DIMM2 memory: 128 Mbyte)
- DIMM3, 2-sided with 128 MBytes on each side (total DIMM3 memory: 256 Mbyte)

A 64-bit interface to DRAM is used (revision CG and earlier revisions).

Because two of the DIMMs are the same size, there are two different ways to program the base address and base mask registers. One way to program them is as follows:

Function 2, Offset 80h = 0000\_2132h // CS0/1 = 128 MB; CS2/3 = 256 MB; CS4/5 = 64 MB; CS6/7 = 128 MB

Function 2, Offset 5Ch = 0380\_0001h // 896 MB base = 768 MB + 128 MB

Function 2, Offset 58h = 0300\_0001h // 768 MB base = 640 MB + 128 MB

Function 2, Offset 54h = 0440\_0001h // 1088 MB base = 1024 MB + 64 MB

Function 2, Offset 50h = 0400\_0001h // 1024 MB base = 896 MB + 128 MB

Function 2, Offset 4Ch = 0100\_0001h // 256 MB base = 0 MB + 256 MB

Function 2, Offset 48h = 0000\_0001h // 0 MB base

Function 2, Offset 44h = 0280\_0001h // 640 MB base = 512 MB + 128 MB

Function 2, Offset 40h = 0200\_0001h // 512 MB base = 256 MB + 256 MB

Function 2, Offset 7Ch = 0060\_FE00h // CS7 = 128 MB

Function 2, Offset 78h = 0060\_FE00h // CS6 = 128 MB

Function 2, Offset 74h = 0020\_FE00h // CS5 = 64 MB

Function 2, Offset 70h = 0020\_FE00h // CS4 = 64 MB

Function 2, Offset 6Ch = 00E0\_FE00h // CS3 = 256 MB

Function 2, Offset 68h = 00E0\_FE00h // CS2 = 256 MB

Function 2, Offset 64h = 0060\_FE00h // CS1 = 128 MB

Function 2, Offset 60h = 0060\_FE00h // CS0 = 128 MB

### 3.5.6.2 DRAM Address Mapping in Interleaving Mode

In memory interleaving mode all DIMM chip-select ranges are the same size and type, and the number of chip selects is a power of two. A BIOS algorithm for programming DRAM CS Base Address and DRAM CS Mask registers in memory interleaving mode is:

1. Program all DRAM CS Base Address and DRAM CS Mask registers using contiguous mapping.
2. For each enabled chip select, swap BaseAddrHi bits with BaseAddrLo bits, defined in Table 13 and Table 14 for revision CG and earlier revisions, and in Table 15 and Table 16 for revision D and later revisions.
3. For each enabled chip select, swap AddrMaskHi bits with AddrMaskLo bits, defined in Table 13 and Table 14 for revision CG and earlier revisions, and in Table 15 and Table 16 for revision D and later revisions.

**Table 13. Swapped physical address lines for interleaving with 64-bit interface (revision CG and earlier revisions)**

Chip Select Mode	Chip Select Size	Swapped Base Address and Address Mask bits		
		8 way interleaving	4 way interleaving	2 way interleaving
000b	32-MB	[27:25] and [15:13]	[26:25] and [14:13]	[25] and [13]
001b	64-MB	[28:26] and [16:14]	[27:26] and [15:14]	[26] and [14]
010b	128-MB	[29:27] and [16:14]	[28:27] and [15:14]	[27] and [14]
011b	256-MB	[30:28] and [17:15]	[29:28] and [16:15]	[28] and [15]
100b	512-MB	[31:29] and [17:15]	[30:29] and [16:15]	[29] and [15]
101b	1-GB	[32:30] and [18:16]	[31:30] and [17:16]	[30] and [16]
110b	2-GB	[33:31] and [18:16]	[32:31] and [17:16]	[31] and [16]

**Table 14. Swapped physical address lines for interleaving with 128-bit interface (revision CG and earlier revisions)**

Chip Select Mode	Chip Select Size	Swapped Base Address and Address Mask bits		
		8 way interleaving	4 way interleaving	2 way interleaving
000b	64-MB	[28:26] and [16:14]	[27:26] and [15:14]	[26] and [14]
001b	128-MB	[29:27] and [17:15]	[28:27] and [16:15]	[27] and [15]
010b	256-MB	[30:28] and [17:15]	[29:28] and [16:15]	[28] and [15]
011b	512-MB	[31:29] and [18:16]	[30:29] and [17:16]	[29] and [16]
100b	1-GB	[32:30] and [18:16]	[31:30] and [17:16]	[30] and [16]
101b	2-GB	[33:31] and [19:17]	[32:31] and [18:17]	[31] and [17]
110b	4-GB	Not implemented.	[33:32] and [18:17]	[32] and [17]

**Table 15. Swapped physical address lines for interleaving with 64-bit interface (revision D and later revisions)**

Chip Select Mode	Chip Select Size	Swapped Base Address and Address Mask bits		
		8 way interleaving	4 way interleaving	2 way interleaving
0000b	32-MB	[27:25] and [15:13]	[26:25] and [14:13]	[25] and [13]
0001b	64-MB	[28:26] and [16:14]	[27:26] and [15:14]	[26] and [14]
0010b	128-MB	[29:27] and [16:14]	[28:27] and [15:14]	[27] and [14]
0011b	128-MB	[29:27] and [17:15]	[28:27] and [16:15]	[27] and [15]
0100b	256-MB	[30:28] and [17:15]	[29:28] and [16:15]	[28] and [15]
0101b	512-MB	[31:29] and [17:15]	[30:29] and [16:15]	[29] and [15]
0110b	256-MB	[30:28] and [18:16]	[29:28] and [17:16]	[28] and [16]
0111b	512-MB	[31:29] and [18:16]	[30:29] and [17:16]	[29] and [16]
1000b	1-GB	[32:30] and [18:16]	[31:30] and [17:16]	[30] and [16]
1001b	1-GB	[32:30] and [19:17]	[31:30] and [18:17]	[30] and [17]
1010b	2-GB	[33:31] and [19:17]	[32:31] and [18:17]	[31] and [17]

**Table 16. Swapped physical address lines for interleaving with 128-bit interface (revision D and later revisions)**

Chip Select Mode	Chip Select Size	Swapped Base Address and Address Mask bits		
		8 way interleaving	4 way interleaving	2 way interleaving
0000b	64-MB	[28:26] and [16:14]	[27:26] and [15:14]	[26] and [14]
0001b	128-MB	[29:27] and [17:15]	[28:27] and [16:15]	[27] and [15]
0010b	256-MB	[30:28] and [17:15]	[29:28] and [16:15]	[28] and [15]



**Table 16. Swapped physical address lines for interleaving with 128-bit interface  
(revision D and later revisions)**

Chip Select Mode	Chip Select Size	Swapped Base Address and Address Mask bits		
		8 way interleaving	4 way interleaving	2 way interleaving
0011b	256-MB	[30:28] and [18:16]	[29:28] and [17:16]	[28] and [16]
0100b	512-MB	[31:29] and [18:16]	[30:29] and [17:16]	[29] and [16]
0101b	1-GB	[32:30] and [18:16]	[31:30] and [17:16]	[30] and [16]
0110b	512-MB	[31:29] and [19:17]	[30:29] and [18:17]	[29] and [17]
0111b	1-GB	[32:30] and [19:17]	[31:30] and [18:17]	[30] and [17]
1000b	2-GB	[33:31] and [19:17]	[32:31] and [18:17]	[31] and [17]
1001b	2-GB	Not implemented.	[32:31] and [19:18]	[31] and [18]
1010b	4-GB	Not implemented.	[33:32] and [19:18]	[32] and [18]

**Example.** DRAM memory consists of two 2-sided DIMMs with 256 MBytes on each side. A 64-bit interface to DRAM is used (revision C and earlier revisions).

1. Register settings for contiguous memory mapping are:

Function 2, Offset 80h = 0000\_0033h // CS0/1 = 256 MB; CS2/3 = 256 MB

Function 2, Offset 40h = 0000\_0001h // 0 MB base

Function 2, Offset 44h = 0100\_0001h // 256 MB base = 0 MB + 256 MB

Function 2, Offset 48h = 0200\_0001h // 512 MB base = 256 MB + 256 MB

Function 2, Offset 4Ch = 0300\_0001h // 768 MB base = 512 MB + 256 MB

Function 2, Offset 60h = 00e0\_fe00h // CS0 = 256 MB

Function 2, Offset 64h = 00e0\_fe00h // CS1 = 256 MB

Function 2, Offset 68h = 00e0\_fe00h // CS2 = 256 MB

Function 2, Offset 6Ch = 00e0\_fe00h // CS3 = 256 MB

2. Base Address bits to be swapped are defined in Table 13 (64-bit interface), 256MByte row (chip select size), 4 way interleaving column (4 chip selects are used). Base Address bits [29:28] are defined with BaseAddrHi[25:24]. Base Address bits [16:15] are defined with BaseAddrLo[12:11].

Function 2, Offset 40h=0000\_0001h

Function 2, Offset 44h=0000\_0801h

Function 2, Offset 48h=0000\_1001h

Function 2, Offset 4Ch=0000\_1801h

3. Address Mask bits to be swapped are the same as the Base Address bits defined in the previous step. Address Mask bits [29:28] are defined with AddrMaskHi[25:24]. Address Mask bits [16:15] are defined with AddrMaskLo[12:11].

Function 2, Offset 60h=03e0\_e600h

Function 2, Offset 64h=03e0\_e600h

Function 2, Offset 68h=03e0\_e600h

Function 2, Offset 6Ch=03e0\_e600h

### 3.5.7 Address to Node/Chip Select Translation

The system address map consists of DRAM memory and memory mapped IO space. A system address is mapped to a node with DRAM base registers (Function 1, Offsets 40h, 48h, etc.) and DRAM limit registers (Function 1, Offsets 44h, 4Ch, etc.). A system address (SysAddr) that is within the address range for a node, and is not a memory mapped I/O address, is normalized (InputAddr) and forwarded to the DRAM controller on that node (see 3.4.4 for details on SysAddr to InputAddr translation). A DRAM address (InputAddr) is mapped to DRAM chip selects with DRAM CS base address registers (Function 2, Offsets 40h, 44h, etc.) and DRAM CS mask registers (Function 2, Offsets 60h, 64h, etc.) (see 3.5.4 and 3.5.5).

The following algorithm is designed to be used to determine the node and the chip select for a system address that maps to DRAM. SystemAddr is a 32 bit input variable that is equal to bits 39-8 of the system address. CSFound, NodeID, and CS are output variables. If CSFound is equal to 1, then NodeID and CS outputs are equal to the node and the chip select that corresponds to the input address.

```
(int,int,int) TranslateSysAddrToCS((uint32)SystemAddr){
int    CSFound, NodeID, CS, F1Offset, F2Offset, Ilog;
uint32 IntlvEn, IntlvSel;
uint32 DramBase, DramLimit, DramEn;
uint32 HoleOffset, HoleEn;
uint32 CSBase, CSLimit, CSMask, CSEn;
uint32 InputAddr, Temp;

CSFound = 0;
for(NodeID = 0; NodeID < 8; NodeID++){
    F1Offset = 0x40 + (NodeID << 3);
    DramBase = Get_PCI(bus0, dev24 + NodeID, func1, F1Offset);
    DramEn = DramBase & 0x00000003;
    IntlvEn = (DramBase & 0x00000700) >> 8;
    DramBase = DramBase & 0xFFFF0000;
    DramLimit = Get_PCI(bus0, dev24 + NodeID, func1, F1Offset + 4);
    IntlvSel = (DramLimit & 0x00000700) >> 8;
    DramLimit = DramLimit | 0x0000FFFF;
    HoleEn = Get_PCI(bus0, dev24 + NodeID, func1, 0xF0);
    HoleOffset = (HoleEn & 0x0000FF00) << 8;
    HoleEn = (HoleEn & 0x00000001);
    if(DramEn && DramBase <= SystemAddr && SystemAddr <= DramLimit){
        if(IntlvEn){
            if(IntlvSel == ((SystemAddr >> 4) & IntlvEn)){
                if(IntlvEn == 1) Ilog = 1;
                else if(IntlvEn == 3) Ilog = 2;
                else if(IntlvEn == 7) Ilog = 3;
            }
        }
    }
}
```

```

        else break;
        Temp = (SystemAddr >> (4 + Ilog)) << 4;
        InputAddr = (Temp | (SystemAddr & 0x0000000F)) << 4;
    }
    else continue;
}
else{
    InputAddr = (SystemAddr - DramBase) << 4;
}
If(HoleEn && SystemAddr > 0x00FFFFFFF)
    InputAddr = InputAddr - HoleOffset;
for(CS = 0; CS < 8; CS++){
    F2Offset = 0x40 + (CS << 2);
    CSBase = Get_PCI(bus0, dev24 + NodeID, func2, F2Offset);
    CSEn = CSBase & 0x00000001;
    CSBase = CSBase & 0xFFE0FE00;
    CSMask = Get_PCI(bus0, dev24 + NodeID, func2, F2Offset + 0x20);
    CSMask = (CSMask | 0x001F01FF) & 0x3FFFFFFF;
    if(CSEn && ((InputAddr & ~CSMask) == (CSBase & ~CSMask))){
        CSFound = 1;
        break;
    }
}
}
if(CSFound) break;
}
return(CSFound, NodeID, CS);
}

```

### 3.5.8 Memory Hoisting

Typically, memory mapped I/O space is mapped from MMIOLowAddr to 4GB-1, and DRAM memory is mapped from address 0 to TOP\_MEM, an address that is less or equal to MMIOLowAddr-1. DRAM memory can also be mapped from 4GB to TOM2, an address greater than 4GB. Memory hoisting is defined as reclaiming the DRAM space that would naturally reside in the MMIO hole just below the 4G address level.

#### 3.5.8.1 Memory Hoisting For Revision D and Earlier Revisions

Depending on the MMIOLowAddr value and how DRAM is populated, DRAM base and limit registers and DRAM CS base and limit registers need to be programmed differently. If possible, contiguous memory mapping is used. Sometimes a memory hole is generated in the DRAM memory map (node hoisting) or in the DRAM CS map (chip select hoisting). In the case of chip select hoisting, the memory hole is included in the memory map for the node that has a hoisted chip select. The memory hole generated by the BIOS is equal to or greater than 4GB-MMIOLowAddr.

Chip select hoisting can not be used in the case of a single 4GB chip select bank. Also, DRAM and node memory interleaving can not be used on a node with chip select hoisting.

MMIOLowAddr is not known by the BIOS at the time when DRAM chip selects are programmed in POST. The BIOS either assumes the MMIOLowAddr value or uses the value selected by the user with a setup option.

The following algorithm is designed to be used to generate the DRAM memory map and DRAM CS map. TOP\_MEM is MSR C001\_001Ah, TOM2 is C001\_001Dh, MAX\_NODES is number of nodes in the system, NodeBase[i] are DRAM base registers, CSBase[i] are DRAM CS base registers, CSMask[i] are DRAM CS mask registers, BE[i] are CSBE bits in DRAM CS base registers.

```
TOP_MEM = TOM2 = MemTop = 0;
for(Node = 0; Node < MAX_NODES; Node++){
    NextCSBase = 0;
    NodeBase[Node] = MemTop;
    for(p = 0; p < 8; p++){
        MaxBankSize = -1;
        for(q = 0; q < 8; q++){
            if(Size[q] != 0 && BE[q] == 0 && Size[q] > MaxBankSize){
                MaxBankSize = Size[q];
                b = q;
            }
        }
        if(MaxBankSize != -1){
            if(MemTop && MemTop < 4GB && (MemTop + Size[b]) > MMIOLowAddr){
                TOP_MEM = MemTop;
                MemTop = 4GB;
                if(NodeBase[Node] == TOP_MEM){ // Node hoisting
                    NodeBase[Node] = 4GB;
                    NextCSBase = 0;
                }
                else { // Chip select hoisting
                    CSBase[b] = 4GB - NodeBase[Node];
                    NextCSBase = 4GB - NodeBase[Node];
                }
            }
            else {
                CSBase[b] = NextCSBase;
            }
            MemTop = MemTop + Size[b];
            CSMask[b] = Size[b] - 1;
            NextCSBase = NextCSBase + Size[b];
            BE[b] = 1;
        }
    }
}
if (TOP_MEM) TOM2 = MemTop;
else TOP_MEM = MemTop;
```

### 3.5.8.2 Memory Hoisting For Revision E and Later Revisions

Memory that resides in the MMIO is repositioned above the 4G level by programming the DRAM Hole Address Register and the DramLimit properly.

The memory hoisting offset field, `DramHoleOffset`, is programmed as shown in the following equation:

$$\text{DramHoleOffset}[31:24] = ( (100\text{h} - \text{DramHoleBase}[31:24]) + \text{DramBase}[31:24] );$$

The `DramLimit` field must be programmed to include the size of the DRAM hole as shown in the following equation:

$$\text{DramLimit} = 4\text{GB} + \text{Hole Size};$$

### 3.5.8.2.1 Non-Node-Interleave Mode

For a system that is configured in the non-node-interleave mode, the DRAM Hole Address Register is enabled only on the node that has its DRAM space overlapped with the MMIO hole if the calculated `DramHoleOffset` is not equal to 4GB. The `DramLimit` of the current node must be adjusted by adding the size of the MMIO hole. The `DramBase` and `DramLimit` registers for the following nodes must also be adjusted by adding the size of the MMIO hole.

**Example.** A two processor system with 4 GB DRAM on each processor and a 1 GB MMIO hole.

```
Processor 0 DramBase[39:24] = 0000h
Processor 0 DramLimit[39:24] = 013Fh (5GB -1; DramLimit + Hole Size)
Processor 0 DramHoleBase[31:24] = C0h (3GB; Starting address of MMIO hole)
Processor 0 DramHoleOffset [31:24] = 40h (1GB)
Processor 1 DramHoleValid = 1(Memory hoisting enabled)
Processor 1 DramBase[39:24] = 0140h (5GB)
Processor 1 DramLimit[39:24] = 023Fh (9GB -1)
Processor 1 DramHoleBase[31:24] = 00h (Memory hoisting is not enabled on node 1)
Processor 1 DramHoleOffset [31:24] = 00b (Memory hoisting is not enabled on node 1)
Processor 1 DramHoleValid = 0 (Memory hoisting disabled)
```

If the calculated `DramHoleOffset` is equal to 4GB, meaning the `DramBase` of the current node is aligned to the starting address of the MMIO hole, the DRAM Hole Address Register must not be enabled. Both the `DramBase` and `DramLimit` of current node, and for the following nodes, must be adjusted by adding the size of the MMIO hole.

**Example.** A two processor system with 2GB memory on each processor and a 2GB MMIO hole.

```
Processor 0 DramBase[39:24] = 0000h
Processor 0 DramLimit[39:24] = 007Fh (2GB - 1)
Processor 0 DramHoleBase[31:24] = 00h
Processor 0 DramHoleOffset[31:24] = 00h (calculated DramHoleOffset = 4GB)
Processor 0 DramHoleValid = 0 (Memory hoisting disabled)
Processor 1 DramBase[39:24] = 0100h (4GB; DramBase + hole size )
```

Processor 1 DramLimit[39:24] = 017Fh (6GB - 1; DramLimit + hole size)

Processor 1 DramHoleBase[31:24] = 00h

Processor 1 DramHoleOffset[31:24] = 00h

Processor 1 DramHoleValid = 0 (Memory hoisting disabled)

### 3.5.8.2.2 Node-Interleave Mode

For a system with the node-interleaving enabled, the DRAM Hole Address Register is enabled on every node with the same setting in its DramHoleBase and DramHoleOffset fields. The DramLimit of every node in the DRAM address map must be adjusted by adding the size of the MMIO hole.

**Example.** A two processor system with 2GB memory on each processor and a 2GB MMIO hole with the node-interleave mode enabled.

Processor 0 DramBase[39:24] = 0000h

Processor 0 DramLimit[39:24] = 017Fh (6GB - 1; DramLimit + hole size)

Processor 0 DramHoleBase[31:24] = 80h (2G; Starting address of MMIO hole)

Processor 0 DramHoleOffset[31:24] = 80h (Calculated DramHoleOffset = 2GB)

Processor 0 DramHoleValid = 1

Processor 1 DramBase[39:24] = 0000h (Same as processor 0)

Processor 1 DramLimit[39:24] = 017Fh (Same as processor 0)

Processor 1 DramHoleBase[31:24] = 80h (Same as processor 0)

Processor 1 DramHoleOffset[31:24] = 80h (Same as processor 0)

Processor 1 DramHoleValid = 1

## 3.5.9 DRAM Timing Low Register

This register contains timing parameters specified in a DRAM data sheet.

### DRAM Timing Low Register

### Function 2: Offset 88h

31	29	28	27	26	24	23	20	19	18	16	15	14	12	11	8	7	4	3	2	0
reserved	Twr	reserved	Trp	Tras	reserved	Trrd	reserved	Trcd	Trfc	Trc	reserved	Tcl								

Bits	Mnemonic	Function	R/W	Reset
31–29	reserved		R	0
28	Twr	Write Recovery Time	R/W	0
27	reserved		R	0
26–24	Trp	Row Precharge Time	R/W	0

Bits	Mnemonic	Function	R/W	Reset
23–20	Tras	Minimum RAS# Active Time	R/W	0
19	reserved		R	0
18–16	Trrd	Active-to-active (RAS#-to-RAS#) Delay	R/W	0
15	reserved		R	0
14–12	Trcd	RAS#-active to CAS#-read/write Delay	R/W	0
11–8	Trfc	Row Refresh Cycle Time	R/W	0
7–4	Trc	Row Cycle Time	R/W	0
3	reserved		R	0
2–0	Tcl	CAS# Latency	R/W	0

## Field Descriptions

**CAS# Latency (Tcl)**—Bits 2–0. Specifies the CAS#-to-read-data-valid.

000b = reserved  
 001b = CL=2  
 010b = CL=3  
 011b = reserved  
 100b = reserved  
 101b = CL=2.5  
 110b = reserved  
 111b = reserved

**Row Cycle Time (Trc)**—Bits 7–4. RAS#-active to RAS#-active or auto refresh of the same bank. Typically ~70 ns. These bits are encoded as follows (only 7–13 are valid; all others are reserved):

0000b = 7 bus clocks  
 0001b = 8 bus clocks  
 ...  
 1110b = 21 bus clocks  
 1111b = 22 bus clocks

**Row Refresh Cycle Time (Trfc)**—Bits 11–8. Auto-refresh-active to RAS#-active or RAS# auto-refresh.

0000b = 9 bus clocks  
 0001b = 10 bus clocks  
 ...  
 1110b = 23 bus clocks  
 1111b = 24 bus clocks

**RAS#-active to CAS#-read/write Delay (Trcd)**—Bits 14–12. Specifies the RAS#-active to CAS#-read/write delay to the same bank.

000b = reserved  
 001b = reserved

010b = 2 bus clocks  
 011b = 3 bus clocks  
 100b = 4 bus clocks  
 101b = 5 bus clocks  
 110b = 6 bus clocks  
 111b = reserved

**Active-to-active (RAS#-to-RAS#) Delay (Trrd)**—Bits 18–16. Specifies the active-to-active delay (RAS#-to-RAS#) of different banks.

000b = reserved  
 001b = reserved  
 010b = 2 bus clocks  
 011b = 3 bus clocks  
 100b = 4 bus clocks  
 101b = reserved  
 110b = reserved  
 111b = reserved

**Minimum RAS# Active Time (Tras)**—Bits 23–20. Specifies the minimum RAS# active time.

0000b to 0100b = reserved  
 0101b = 5 bus clocks  
 ...  
 1111b = 15 bus clocks

**Row Precharge Time (Trp)**—Bits 26–24. Specifies the Row precharge Time. (Precharge-to-Active or Auto-Refresh of the same bank.

000b = reserved  
 001b = reserved  
 010b = 2 bus clocks  
 011b = 3 bus clocks  
 100b = 4 bus clocks  
 101b = 5 bus clocks  
 110b = 6 bus clocks  
 111b = reserved

**Write Recovery Time (Twr)**—Bit 28. Measures when the last write datum is safely registered by the DRAM. It measures from the last data to precharge (writes can go back-to-back).

0 = 2 bus clocks  
 1 = 3 bus clocks

### 3.5.10 DRAM Timing High Register

This register contains the normal timing parameters specified in a DRAM data sheet.



## DRAM Timing High Register

## Function 2: Offset 8Ch

31	23	22	20	19	13	12	8	7	6	4	3	1	0
reserved		Twcl		reserved		Tref		reserved	Trwt		reserved		Twtr

Bits	Mnemonic	Function	R/W	Reset
31–23	reserved		R	0
22–20	Twcl	Write CAS Latency	R/W	0
19–13	reserved		R	0
12–8	Tref	Refresh Rate	R/W	0
7	reserved		R	0
6–4	Trwt	Read-to-Write Delay	R/W	0
3–1	reserved		R	0
0	Twtr	Write-to-Read Delay	R/W	0

## Field Descriptions

**Write-to-Read Delay (Twtr)**—Bit 0. This bit specifies the write-to-read delay. It is measured from the rising edge following the last non-masked data strobe to the rising edge of the next Read command.

0 = 1 bus Clocks

1 = 2 bus Clocks

**Read-to-Write Delay (Trwt)**—Bits 6–4. Specifies the read-to-write delay.

This is not a DRAM-specified timing parameter, but must be considered due to routing latencies on the clock forwarded bus. It is counted from the first address bus slot that was not associated with part of the read burst. These bits are encoded as follows:

000b = 1 bus clocks

001b = 2 bus clocks

010b = 3 bus clocks

011b = 4 bus clocks

100b = 5 bus clocks

101b = 6 bus clocks

110b = reserved

111b = reserved

**Refresh Rate (Tref)**—Bits 12–8. Specifies the refresh rate of the DIMM requiring the most frequent refresh. These bits are encoded as follows:

00000b = 100 MHz 15.6us

00001b = 133 MHz 15.6us

00010b = 166 MHz 15.6us

00011b = 200 MHz 15.6us

01000b = 100 MHz 7.8us  
01001b = 133 MHz 7.8us  
01010b = 166 MHz 7.8us  
01011b = 200 MHz 7.8us  
10000b = 100 MHz 3.9us  
10001b = 133 MHz 3.9us  
10010b = 166 MHz 3.9us  
10011b = 200 MHz 3.9us

**Write CAS Latency (Twcl)**—Bits 22–20. Specifies the write CAS latency. Unbuffered DIMMs should be programmed as 0 and registered DIMMs as 1. These bits are encoded as follows:

000b = 1 Mem clock after CAS#  
001b = 2 Mem clocks after CAS#

### 3.5.11 DRAM Configuration Registers

The following registers contain all the bits to configure the DRAMs for proper operation.

#### 3.5.11.1 DRAM Configuration Low Register

This register controls drive strength, refresh, and initialization.

##### DRAM Configuration Low Register Function 2: Offset 90h

Bits	Mnemonic	Function	R/W	Reset
31–30	PwrDownCtl	Power Down Control	R/W	0
29	UpperCSMap	Upper Chip Select Mapping	R/W	0
28	En2T	Enable 2T Timing	R/W	0
27–25	BypMax	Bypass Max	R/W	0
24	DisInRcvrs	Disable DRAM Receivers	R/W	0
23–20	x4DIMMS	x4DIMMS	R/W	0
19	32ByteEn	Enable 32-Byte Granularity	R/W	0
18	UnBuffDimm	Unbuffered DIMMs	R/W	0
17	DimmEcEn	DIMM ECC Enable	R/W	0
16	128/64	128-Bit/64-Bit	R/W	0
15–14	RdWrQByp	Read/Write Queue Bypass Count	R/W	0
13	SR_S	Self-Refresh Status	R/W	0
12	ESR	Exit Self-Refresh	R/W	0
11	MemClrStatus	Memory Clear Status	R	0
10	DramEnable	DRAM Enable	R	0
9	DualDimmEn	Dual DIMM Enable	R/W	0
8	DramInit	DramInit	R/W	0
7	ScratchBit	BIOS Scratch Bit revision D and earlier revisions	R/W	0
7	PwrDwnTriEn	Power Down Tristate Enable revision E	R/W	0
6	Mod64BitMux	Mismatched DIMM Support Enable	R/W	0

Bits	Mnemonic	Function	R/W	Reset
5	Burst2Opt	DRAM Burst of 2	R/W	0
4	reserved		R	0
3	DisDqsHys	Disable DQS Hysteresis	R/W	0
2	QFC_EN	QFC_EN Enable	R/W	0
1	D_DRV	D_DRV	R/W	0
0	DLL_Dis	DLL Disable	R/W	0

## Field Descriptions

**DLL Disable (DLL\_Dis)**—Bit 0. Disables the Digital Locked Loop that controls relationship between the memory Data and the memory strobes (DQS).

0 = Enabled (default)  
1 = Disabled

**D\_DRV (D\_DRV)**—Bit 1. Controls the drive strength of the DRAM device outputs when the DRAM is driving on a READ command. This bit is required by the DRAM in the Extended Memory Register Set (EMRS) command. It applies to all outputs.

0 = Normal Drive (default)  
1 = Weak Drive (beware)

**QFC\_EN Enable (QFC\_EN)**—Bit 2. Causes the corresponding enable bit to be set in the (external) DRAM extended mode register. The QFC signal is an optional DRAM output control used to isolate module loads (DIMMs) from the system memory bus by means of FET switches when the given module is not being accessed.

0 = Disabled (default)  
1 = Enabled

**Disable DQS Hysteresis (DisDqsHys)**—Bit 3. Disables DQS input filtering when set to 1. It is required that BIOS (1) set this bit high prior to initializing DRAM (through bit 8, DramInit) and (2), in a subsequent access to this register, clear this bit; the write that clears this bit may be concurrent with the write that sets DramInit or it may be prior to the write that sets DramInit.

**DRAM Burst of 2 (Burst2Opt)**—Bit 5. This bit enables DRAM burst of 2 when set to 1. This bit only has an effect if the DRAM controller is configured for a 128-bit interface, 32 byte granularity, unbuffered DIMMs and 1T timing. This bit is only valid for processors in the 939 package. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Mismatched DIMM Support Enable (Mod64BitMux)**—Bit 6. This bit enables support for mismatched DIMMs when using 128-bit DRAM interface. When this bit is set the UpperCSMap bit and the 128/64 bit have no effect. This bit only applies to the 939 package. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™

Processors” on page 29 for revision information about this field. Revision D and earlier revisions

**BIOS Scratch Bit (SctatchBit)**—Bit 7. This bit controls no hardware and can be used as a scratch bit by BIOS. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Power Down Tristate Enable (PwrDwnTriEn)**—Bit 7. This bit enables tristating the DDR address and command bus when their associated DIMMs are in the DDR power down state. This function is only enabled when this bit is set and the DRAM controller is configured in one of the power down modes as specified by the PwrDownCtl bits. This bit should never be set when using registered DIMMs. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**DramInit (DramInit)**—Bit 8. Set by the BIOS to initiate the DRAM initialization sequence. It is cleared by the DRAM controller when the initialization completes. Bits 3–0 and Tc1 must be properly programmed before (or at same time as) the bit is set to 1. See bit 3 of this register, DisDqsHys, for BIOS programming requirements related to DramInit.  
0 = Initialization done or not started (default)

**Dual DIMM Enable (DualDimmEn)**—Bit 9. When this bit is set, the A copy of the memory address bus is enabled, regardless of the MC0\_EN (Function 2, Offset 94h) value, and the B copy of the memory bus is disabled if 939 package with 128-bit bus is not used. This bit should be set if unbuffered DIMMs are used, and two DIMM sockets are connected to the A copy of the memory address bus, as in SODIMM or 939 package configurations. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**DRAM Enable (DramEn)**—Bit 10. When set, this bit indicates that DRAM is enabled. This bit is set by hardware at the completion of DRAM initialization or on an exit from self refresh. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Memory Clear Status (MemClrStatus)**—Bit 11. When set, this bit indicates that the memory clear function is complete. It is only cleared by reset. BIOS should not write or read the DRAM or enable memory hoisting (Function 1, Offset F0h bit 0) until this bit is set by hardware. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Exit Self-Refresh (ESR)**—Bit 12. BIOS sets either the DramInit bit (bit 8) or the Exit Self-Refresh bit (bit 12) after reset, depending on whether the part is coming out of Suspend-to-RAM mode or not (bit 12 is set when coming out of suspend-to-RAM mode). After this bit is set, it reads back as a 1 until the process of exiting self refresh is complete, at which point it reads back as a 0.

**Self-Refresh Status (SR\_S)**—Bit 13. When set, indicates that the DRAMs are in self-refresh mode.

When it deasserts, indicates that the memory controller is ready to process requests again.

This bit may be polled until the DRAMs exit the self-refresh state. This bit cannot accurately reflect that the DRAM is in self-refresh when resuming from suspend-to-RAM (STR) because the processor loses power in the STR state. Therefore, this bit must be set, along with bit 12, to properly exit suspend-to-RAM mode.

0 = Normal operation (default)

1 = Self-refresh mode active

**Read/Write Queue Bypass Count (RdWrQByp)**—Bits 15–14. Specifies the number of times the oldest operation in the DCI read/write queue can be bypassed before the arbiter is overridden and the oldest operation is chosen.

00b = 2

01b = 4

10b = 8

11b = 16

**128-Bit/64-Bit (128/64)**—Bit 16. Indicates a 128-bit interface to DRAM.

0 = 64-bit interface to DRAM

1 = 128-bit interface to DRAM

**DIMM ECC Enable (DimmEcEn)**—Bit 17. When DimmEcEn is set to 1, it indicates that all enabled DIMMs have support for ECC check bit storage.

0 = Some DIMMs do not have ECC bits

1 = All DIMMs have ECC bits

**Unbuffered DIMMs (UnBuffDimm)**—Bit 18. When set, the DRAM controller is only connected to unbuffered DIMMs. In this case the controller drives address/control 3/4 MEMCLK earlier than the clock to account for the address flight time. If deasserted, the controller drives address/control 1/2 MEMCLK earlier than clock.

0 = Buffered DIMMs

1 = Unbuffered DIMMs

**Enable 32-Byte Granularity (32ByteEn)**—Bit 19. When set, indicates that the burst counter should be chosen to optimize data bus bandwidth for 32-byte accesses (except when the bus width is 128 bits). This bit, along with bit 16, produces the following truth table.

Bit 19	Bit 16	
0	0	8-beat bursts (64 bytes)
0	1	4-beat bursts (64 bytes)
1	0	4-beat bursts (32 bytes)

Bit 19	Bit 16	
1	1	4-beat bursts (64 bytes) 2-beat bursts (32 bytes) for Revision E and later revisions if Burst2Opt = 1, UnBuffDimm = 1, and En2T = 0

**x4DIMMS[3:0]**—Bits 23–20. Indicates whether each of the four DIMMs are x4 or not. Each bit corresponds to a chip select pair (bit 20 corresponds to CS1/0, etc.).

0 = DIMM is not x4

1 = x4 DIMM present

**Disable DRAM Receivers (DisInRcvrs)**—Bit 24. When the memory controller is disabled, this bit is asserted and all DRAM receivers are disabled. The inputs/bidirectional DRAM pins can be left unconnected.

0 = Receivers enabled

1 = Receivers disabled

**Bypass Max (BypMax)**—Bits 27–25. Specifies the number of times the oldest entry in DCQ can be bypassed in arbitration before the arbiter's choice is vetoed. When programmed to all 0s, this feature is disabled, and the choice of the arbiter is always respected.

**Enable 2T Timing (En2T)**—Bit 28. When this bit is set, DRAM commands and address will be driven for 2 clock cycles and qualified with an associated chip select on the second phase of the 2 clock command and address. This bit should only be set with unbuffered DIMMs. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Upper Chip Select Mapping (UpperCSMap)**—Bit 29. When this bit is set, chip select pins CS[7:4] are copies of chip select pins CS[3:0]. This bit should be set if the 939 package is used. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Power Down Control (PwrDownCtl)**—Bits 31–30. This field is initialized based on how the clock enable pins are connected to DIMMs on the board. Power down functionality is added to these pins in addition to initialization and self refresh mode functionality. A DIMM or a group of DIMMs enters power down mode by deasserting the corresponding clock enable signal when the DRAM controller detects that there are no transactions scheduled to any of the DIMMs connected to the clock enable signal. A DIMM or a group of DIMMs exits power down mode by asserting the corresponding clock enable signal when a transaction is scheduled to any DIMM connected to the clock enable signal. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

- 00b=Power down is disabled.
- 01h=Alternating DIMM clock enable control.

Clock Enable	754 Chip Selects	939 Chip Selects	940 Chip Selects
MEMCKEA (754 or 939) MEMCKEC (939) MEMCLK_LO (940)	MEMCS_L[0,2,4,6]	MEMCS_1L_L[0] MEMCS_2L_L[0] MEMCS_1H_L[0] MEMCS_2H_L[0]	MEMCS_L[0,2,4,6]
MEMCKEB (754 or 939) MEMCKED (939) MEMCLK_HI (940)	MEMCS_L[1,3,5,7]	MEMCS_1L_L[1] MEMCS_2L_L[1] MEMCS_1H_L[1] MEMCS_2H_L[1]	MEMCS_L[1,3,5,7]

- 10h=Same DIMM Clock Enable control.

Clock Enable	754 Chip Selects	939 Chip Selects	940 Chip Selects
MEMCKEA (754 or 939) MEMCKEC (939) MEMCLK_LO (940)	MEMCS_L[0,1,4,5]	MEMCS_1L_L[1:0] MEMCS_1H_L[1:0]	MEMCS_L[0,1,4,5]
MEMCKEB (754 or 939) MEMCKED (939) MEMCLK_HI (940)	MEMCS_L[2,3,6,7]	MEMCS_2L_L[1:0] MEMCS_2H_L[1:0]	MEMCS_L[2,3,6,7]

#### Revision D

- 11h=Independent DIMM clock enable control.

Clock Enable	754 Chip Selects	939 Chip Selects	940 Chip Selects
MEMCKEA (754 or 939) MEMCLK_LO (940)	MEMCS_L[0]	MEMCS_1L_L[0]	MEMCS_L[0]
MEMCKEB (754 or 939) MEMCLK_HI (940)	MEMCS_L[1]	MEMCS_1L_L[1]	MEMCS_L[1]
MEMCKEC (939)		MEMCS_2L_L[0]	
MEMCKED (939)		MEMCS_2L_L[1]	
Power down mode is not entered if inactivity is detected on DIMMs selected by	MEMCS_L[7:2]	MEMCS_1H_L[1:0] MEMCS_2H_L[1:0]	MEMCS_L[7:2]

#### Revision E and later revisions

- 11h=Independent DIMM clock enable control.

Clock Enable	754 Chip Selects	939 Chip Selects	940 Chip Selects
MEMCKEA (754 or 939) MEMCLK_LO (940)	MEMCS_L[0]	MEMCS_1L_L[0]	MEMCS_L[0]
MEMCKEB (754 or 939) MEMCLK_HI (940)	MEMCS_L[1]	MEMCS_1L_L[1]	MEMCS_L[1]
MEMCKEC (939)		MEMCS_1H_L[0]	
MEMCKED (939)		MEMCS_1H_L[1]	
Power down mode is not entered if inactivity is detected on DIMMs selected by	MEMCS_L[7:2]	MEMCS_2L_L[1:0] MEMCS_2H_L[1:0]	MEMCS_L[7:2]

### 3.5.11.2 DRAM Configuration High Register

#### DRAM Configuration High Register

#### Function 2: Offset 94h

31	30	29	28	27	26	25	24	23	22	20	19	18	16	15	14	13	12	11	8	7	4	3	0
OddDivisorCorrect	reserved	MC3_EN	MC2_EN	MC1_EN	MC0_EN	MCR	reserved		MemClk	DCC_EN		ILD_lmt		Disable Jitter		MemDQDrvStren	reserved		RdPreamble	reserved		AsyncLat	

Bits	Mnemonic	Function	R/W	Reset
31	OddDivisorCorrect	Odd Divisor Correct	R/W	0
30	reserved		R	0
29	MC3_EN	Memory Clock 3 Enabled	R/W	0
28	MC2_EN	Memory Clock 2 Enabled	R/W	0
27	MC1_EN	Memory Clock 1 Enabled	R/W	0
26	MC0_EN	Memory Clock 0 Enabled	R/W	0
25	MCR	Memory Clock Ratio Valid	R/W	0
24–23	reserved		R	0
22–20	MemClk	DRAM MEMCLK Frequency	R/W	0
19	DCC_EN	Dynamic Idle Cycle Counter Enable	R/W	0
18–16	ILD_lmt	Idle Cycle Limit	R/W	0
15	DisableJitter	Disable Jitter	R/W	0
14–13	MemDQDrvStren	Memory DQ Drive Strength	R/W	
12	reserved		R	0



Bits	Mnemonic	Function	R/W	Reset
11–8	RdPreamble	Read Preamble	R/W	0
7–4	reserved		R	0
3–0	AsyncLat	Maximum Asynchronous Latency	R/W	0

## Field Descriptions

**Maximum Asynchronous Latency (AsyncLat)**—Bits 3–0. This field should be loaded with a 4-bit value equal to the maximum asynchronous latency in the DRAM read round-trip loop.

0000b = 0 ns

...

1111b = 15 ns

**Read Preamble (RdPreamble)**—Bits 11–8. The time prior to the max-read DQS-return when the DQS receiver should be turned on. This is specified in units of 0.5 ns. The controller needs to know when to enable its DQS receiver in anticipation of the DRAM DQS driver turning on for a read. The controller will disable its DQS receiver until the read preamble time and then enable its DQS receiver while the DRAM asserts DQS.

0000b = 2.0 ns

0001b = 2.5 ns

0010b = 3.0 ns

0011b = 3.5 ns

0100b = 4.0 ns

0101b = 4.5 ns

0110b = 5.0 ns

0111b = 5.5 ns

1000b = 6.0 ns

1001b = 6.5 ns

1010b = 7.0 ns

1011b = 7.5 ns

1100b = 8.0 ns

1101b = 8.5 ns

1110b = 9.0 ns

1111b = 9.5 ns

**Memory DQ Drive Strength (MemDQDrvStren)**—Bits 14–13. This field controls the drive strength reduction of the Memory DQ pins. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

00b = Drive strength not reduced.

01b = Approximately 15% reduction

10b = Approximately 30% reduction

11b = Approximately 50% reduction

**Disable Jitter (DisableJitter)**—Bit 15. When set the DDR compensation circuit will not change values unless the change is more than one step from the current value. This bit should be set for all revision E and later processors. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Idle Cycle Limit (ILD\_lmt)**—Bits 18–16. Specifies the number of MemCLKs before forcibly closing (precharging) an open page. If DCC\_EN (Function 2, Offset 94h) has a value of 0, the static counters are loaded with the ILD\_lmt and decremented each clock. If DCC\_EN (Function 2, Offset 94h) has a value of 1, the dynamic counters are loaded with the ILD\_lmt and modified as follows:

**Increment**—When a Page Miss (PM) page hits on an invalid entry in the Page Table. The presumption is that in the past, that page table entry was occupied by the very same page that has a Page Miss. Had the old page been kept open longer, it would have been a Page Hit. Increment the Idle Cycle Limit count to increase the probability of getting a future Page Hit.

**Decrement**—When a Page Conflict (PC) arrives and hits on an idle entry (obviously an open page). This Page Conflict can be avoided if the open page is closed earlier. Decrement the Idle Cycle Limit count to increase the probability of avoiding a future Page Conflict.

000b = 0 cycles  
 001b = 4 cycles  
 010b = 8 cycles  
 011b = 16 cycles  
 100b = 32 cycles  
 101b = 64 cycles  
 110b = 128 cycles  
 111b = 256 cycles

**Dynamic Idle Cycle Counter Enable (DCC\_EN)**—Bit 19. When set to 1, indicates that each entry in the page table dynamically adjusts the idle cycle limit based on Page Conflict/Page Miss (PC/PM) traffic.

**DRAM MEMCLK Frequency (MemClk)**—Bits 22–20. Specifies the DRAM MEMCLK frequency in MHz. It is possible to program this field for a higher frequency than the maximum allowed by the processor. Refer to the processor data sheet for the maximum operating frequency allowed for a given processor implementation.

000b = 100 MHz  
 001b = reserved  
 010b = 133 MHz  
 011b = reserved  
 100b = reserved  
 101b = 166 MHz  
 110b = reserved  
 111b = 200 MHz

**Memory Clock Ratio Valid (MCR)**—Bit 25. Indicates to the controller to drive MEMCLK. The following fields need to be programmed before this bit is set: MemClk (Function 2, Offset 94h), MCi\_EN (Function 2, Offset 94h), UnBuffDimm (Function 2, Offset 90h), and 128/64 (Function 2, Offset 90h). This bit is also set after the Base Address registers have been configured. This allows the system to know if a CS is occupied before clocks start toggling. (The intent is to not drive clocks to unoccupied slots to reduce the potential for EMI.)

0 = Disable MemCLKs (default)

1 = Enable MemCLKs

**Memory Clock 0 Enabled (MC0\_EN)**—Bit 26. The MC0\_EN bit is set to 1 to enable operation of the MEMCLK signals to DIMM0.

0 = Disabled (default)

1 = Enabled

**Memory Clock 1 Enabled (MC1\_EN)**—Bit 27. The MC1\_EN bit is set to 1 to enable operation of the MEMCLK signals to DIMM1.

0 = Disabled (default)

1 = Enabled

**Memory Clock 2 Enabled (MC2\_EN)**—Bit 28. The MC2\_EN bit is set to 1 to enable operation of the MEMCLK signals to DIMM2.

0 = Disabled (default)

1 = Enabled

**Memory Clock 3 Enabled (MC3\_EN)**—Bit 29. The MC3\_EN bit is set to 1 to enable operation of the MEMCLK signals to DIMM3.

0 = Disabled (default)

1 = Enabled

**Odd Divisor Correct (OddDivisorCorrect)**—Bit 31. When set, the memory controller eliminates odd divisors from being used in the generation of the memory clock. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

### 3.5.12 DRAM Delay Line Register

The DRAM Delay Line register is used to adjust the skew of the input DQS strobe relative to the data.

## DRAM Delay Line Register

## Function 2: Offset 98h

31	28	27	26	25	24	23	16	15	0
reserved	DllSpeedOverride	DllSpeed	CompPwrSaveEn	AltVidC3MemClkTriEn	reserved	AdjFaster	AdjSlower	Adj	reserved

Bits	Mnemonic	Function	R/W	Reset
31	Reserved		R	0
30	DllSpeedOverride	DLL Speed Override	W	0
29	DllSpeed	DLL Speed	W	0
28	CompPwrSaveEn	DDR Compensation Power Save Enable	R/W	0
27	AltVidC3MemClkTriEn	AltVID C3 Memory Clock Tristate Enable	R/W	0
26	reserved		R	0
25	AdjFaster	Adjust Faster	R/W	0
24	AdjSlower	Adjust Slower	R/W	0
23–16	Adj	Delay Line Adjust	R/W	0
15–0	reserved		R	0

## Field Descriptions

**Delay Line Adjust (Adj)**—Bits 23–16. Adjusts the DLL derived PDL delay by one or more delay stages in the faster or slower direction.

**Adjust Slower (AdjSlower)**—Bit 24. Indicates that the value in Adj is used to increase the PDL delay.

**Adjust Faster (AdjFaster)**—Bit 25. Indicates that the value in Adj is used to decrease the PDL delay.

**AltVID C3 Memory Clock Tristate Enable (AltVidC3MemClkTriEn)**—Bit 27. Enables the DDR memory clocks to be tristated when alternate VID mode is enabled. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**DDR Compensation Power Save Enable (CompPwrSaveEn)**—Bit 28. This bit enables a power savings mode by disabling the DDR I/O compensation circuitry when the DRAM is in self refresh. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**DLL Speed (DllSpeed)**—Bit 29. If DllSpeedOverride=1, the D200\_L control signal to the DLL is forced to the value of this bit. 1=High speed. 0=Low speed. See “Register Differences in

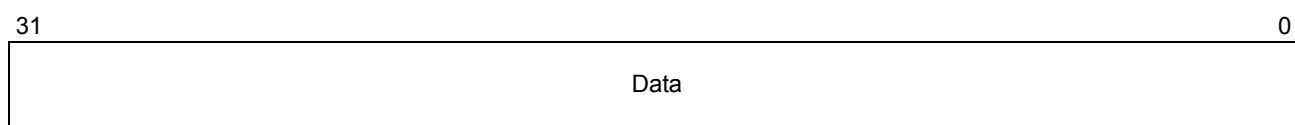
Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**DLL Speed Override (DllSpeedOverride)**—Bit 30. When this bit has a value of 1, override for D200\_L control signal to DLL is enabled. When this bit has a value of 0, then the DLL is set to high speed when MemClk (Function 2, Offset 94h) has a value of 110b or 111h. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

### 3.5.13 Scratch Register

#### Scratch Register

Function 2: Offset 9Ch



Bits	Mnemonic	Function	R/W	Reset
31–0	Data	Scratch Data	R/W	0h

#### Field Descriptions

**Scratch Data (Data)**—Bits 31–0.

## 3.6 Function 3—Miscellaneous Control

This function is a collection of the remaining memory system configuration registers. As listed in Table 17, Function 3 includes the following registers:

- Machine check architecture for Northbridge errors
- DRAM scrubber
- Buffer counts for Northbridge flow control
- Power management
- GART
- Clocking/FIFO control
- Thermtrip status
- Northbridge capabilities

**Table 17. Function 3 Configuration Registers**

Offset	Register Name				Reset	Access	Description
00h	Device ID		Vendor ID (AMD)		1103_1022h	RO	page 119
04h	Status <sup>1</sup>		Command		0000_0000h	RO	
08h	Base Class Code	Subclass Code	Program Interface	Revision ID	0600_0000h	RO	page 120
0Ch	BIST	Header Type	Latency Timer	Cache Line Size	0080_0000h	RO	page 120
10h	Base Address 0				0000_0000h	RO	
14h	Base Address 1				0000_0000h	RO	
18h	Base Address 2				0000_0000h	RO	
1Ch	Base Address 3				0000_0000h	RO	
20h	Base Address 4				0000_0000h	RO	
24h	Base Address 5				0000_0000h	RO	
28h	Card Bus CIS Pointer				0000_0000h	RO	
2Ch	Sub System ID		Sub system Vendor ID		0000_0000h	RO	
30h	ROM Base Address				0000_0000h	RO	
34h	Capabilities				0000_0000h	RO	
38h	reserved				0000_0000h	RO	
3Ch	Max Latency	Min GNT	Int Pin	Int Line	0000_0000h	RO	
40h	MCA Northbridge Control				0000_0000h	RW	page 121
44h	MCA Northbridge Configuration				0000_0000h	RW	page 124
48h	MCA Northbridge Status Low				page 127	RW	page 127
4Ch	MCA Northbridge Status High				page 130	RW	page 130
50h	MCA Northbridge Address Low				page 133	RW	page 133
54h	MCA Northbridge Address High				page 133	RW	page 133
58h	Scrub Control				0000_0000h	RW	page 140
5Ch	DRAM Scrub Address Low				page 142	RW	page 142
60h	DRAM Scrub Address High				page 142	RW	page 142
70h	SRI-to-XBAR Buffer Counts				5102_0111h	RW	page 147
74h	XBAR-to-SRI Buffer Counts				5000_8011h	RW	page 150
78h	MCT-to-XBAR Buffer Counts				0800_3800h	RW	page 148

**Notes:**

1. The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.
2. Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.

**Table 17. Function 3 Configuration Registers (Continued)**

Offset	Register Name	Reset	Access	Description
7Ch	Free List Buffer Counts	0000_221Bh	RW	page 149
80h	Power Management Control Low	0000_0000h	RW	page 153
84h	Power Management Control High	0000_0000h	RW	page 153
90h	GART Aperture Control	0000_0000h	RW	page 154
94h	GART Aperture Base	page 155	RW	page 155
98h	GART Table Base	page 156	RW	page 156
9Ch	GART Cache Control	0000_0000h	RW	page 157
D4h	Clock Power/Timing Low	page 157	RW	page 157
D8h	Clock Power/Timing High	page 160	RW	page 160
DCh	HyperTransport™ Read Pointer Optimization	page 161	RW	page 161
E4h	Thermtrip Status	0000_0000h	RO	page 163
E8h	Northbridge Capabilities	0000_0000h	RO	page 165
<b>Notes:</b> <ol style="list-style-type: none"> <li>1. The unimplemented registers in the standard PCI configuration space are implemented as read-only and return 0 if read.</li> <li>2. Reads and writes to unimplemented registers in the extended PCI configuration space will result in unpredictable behavior.</li> </ol>				

### 3.6.1 Device/Vendor ID Register

This register specifies the device and vendor IDs for the Function 3 registers and is part of the standard PCI configuration header.

#### Device/Vendor ID Register

Function 3: Offset 00h

31	16	15	0
DevID		VenID	

Bits	Mnemonic	Function	R/W	Reset
31–16	DevID	Device ID	R	1103h
15–0	VenID	Vendor ID	R	1022h

#### Field Descriptions

**Vendor ID (VenID)**—Bits 15–0. This read-only value is defined as 1022h for AMD.

**Device ID (DevID)**—Bits 31–16. This read-only value is defined as 1103h for the HyperTransport technology configuration function.

### 3.6.2 Class Code/Revision ID Register

This register specifies the class code and revision for the Function 3 registers and is part of the standard PCI configuration header.

#### Class Code/Revision ID Register

Function 3: Offset 08h

31	24	23	16	15	8	7	0	
BCC				SCC		PI		RevID

Bits	Mnemonic	Function	R/W	Reset
31–24	BCC	Base Class Code	R	06h
23–16	SCC	Subclass Code	R	00h
15–8	PI	Programming Interface	R	00h
7–0	RevID	Revision ID	R	00h

#### Field Descriptions

**Revision ID (RevID)**—Bits 7–0.

**Programming Interface (PI)**—Bits 15–8. This read-only value is defined as 00h.

**Sub Class Code (SCC)**—Bits 23–16. This read-only value is defined as 00h.

**Base Class Code (BCC)**—Bits 31–24. This read-only value is defined as 06h for a host bridge device.

### 3.6.3 Header Type Register

This register specifies the header type for the Function 3 registers and is part of the standard PCI configuration header.

#### Header Type Register

Function 3: Offset 0Ch

31	24	23	16	15	8	7	0	
BIST				HType		LatTimer		CLS

Bits	Mnemonic	Function	R/W	Reset
31–24	BIST	BIST	R	00h
23–16	HType	Header Type	R	80h
15–8	LatTimer	Latency Timer	R	00h
7–0	CLS	Cache Line Size	R	00h



## Field Descriptions

**CacheLineSize (CLS)**—Bits 7–0. This read-only value is defined as 00h.

**LatencyTimer (LatTimer)**—Bits 15–8. This read-only value is defined as 00h.

**HeaderType (Htype)**—Bits 23–16. This read-only value is defined as 80h to indicate that multiple functions are present in the configuration header and that the header layout corresponds to a device header as opposed to a bridge header. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**BIST**—Bits 31–24. This read-only value is defined as 00h.

### 3.6.4 Machine Check Architecture (MCA) Registers

These registers are used to configure the Machine Check Architecture (MCA) functions of the on-chip Northbridge (NB) hardware and to provide a method for the Northbridge hardware to report errors in a way compatible with MCA. All of the Northbridge MCA registers, except for the MCA NB Configuration register, are accessible through the MCA-defined MSR method, as well as through PCI configuration space. The MCA NB Configuration register is accessible only through PCI configuration space.

The MCA NB Control register enables MCA reporting of each error checked by the Northbridge. The global MCA error enables must also be set through the global MCA MSRs. The error enables in this register only affect error reporting through MCA. Actions which the Northbridge may take in addition to MCA reporting are enabled through the MCA NB Configuration register.

The MCA NB Status registers and MCA NB Address registers provide status and address information regarding an error which the Northbridge has logged. The values of error-logging registers are maintained through a warm reset allowing software to identify an error source that resulted in system-wide initialization.

#### 3.6.4.1 MCA NB Control Register

## MCA NB Control Register

### Function 3: Offset 40h

31	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	WchDogTmrEn	AtomicRMWEn	GatTbWkEn	TgtAbrtEn	MstrAbrtEn	SyncPkt2En	SyncPkt1En	SyncPkt0En	CrcErr2En	CrcErr1En	CrcErr0En	UnCorrEccEn	CorrEccEn	

Bits	Mnemonic	Function	R/W	Reset
31–13	reserved		R	0
12	WchDogTmrEn	Watchdog Timer Error Reporting Enable	R/W	0

Bits	Mnemonic	Function	R/W	Reset
11	AtomicRMWEn	Atomic Read-Modify-Write Error Reporting Enable	R/W	0
10	GartTblWkEn	GART Table Walk Error Reporting Enable	R/W	0
9	TgtAbtEn	Target Abort Error Reporting Enable	R/W	0
8	MstrAbtEn	Master Abort Error Reporting Enable	R/W	0
7	SyncPkt2En	HyperTransport Link 2 Sync Packet Error Reporting Enable	R/W	0
6	SyncPkt1En	HyperTransport Link 1 Sync Packet Error Reporting Enable	R/W	0
5	SyncPkt0En	HyperTransport Link 0 Sync Packet Error Reporting Enable	R/W	0
4	CrcErr2En	HyperTransport Link 2 CRC Error Reporting Enable	R/W	0
3	CrcErr1En	HyperTransport Link 1 CRC Error Reporting Enable	R/W	0
2	CrcErr0En	HyperTransport Link 0 CRC Error Reporting Enable	R/W	0
1	UnCorrEccEn	Uncorrectable ECC Error Reporting Enable	R/W	0
0	CorrEccEn	Correctable ECC Error Reporting Enable	R/W	0

## Field Descriptions

**Correctable ECC Error Reporting Enable (CorrEccEn)**—Bit 0. Enables MCA reporting of correctable ECC errors which are detected in the Northbridge. Since correctable errors do not result in an MCA exception, the effect of this bit is limited to the setting of the error enable bit in the MCA NB Status High register.

**Uncorrectable ECC Error Reporting Enable (UnCorrEccEn)**—Bit 1. Enables MCA reporting of uncorrectable ECC errors which are detected in the Northbridge. Note that in some cases data may be forwarded to the CPU core prior to checking ECC in which case the check takes place in one of the other error reporting banks.

**HyperTransport Link 0 CRC Error Reporting Enable (CrcErr0En)**—Bit 2. Enables MCA reporting of CRC errors detected on HyperTransport link 0. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a CRC error on an incoming link independent of the state of this bit.

**HyperTransport Link 1 CRC Error Reporting Enable (CrcErr1En)**—Bit 3. Enables MCA reporting of CRC errors detected on HyperTransport link 1. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a CRC error on an incoming link independent of the state of this bit.

**HyperTransport Link 2 CRC Error Reporting Enable (CrcErr2En)**—Bit 4. Enables MCA reporting of CRC errors detected on HyperTransport link 2. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a CRC error on an incoming link independent of the state of this bit.

**HyperTransport Link 0 Sync Packet Error Reporting Enable (SyncPkt0En)**—Bit 5. Enables MCA reporting of HyperTransport sync error packets detected on HyperTransport link 0. The

Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a sync packet on an incoming link independent of the state of this bit.

**HyperTransport Link 1 Sync Packet Error Reporting Enable (SyncPkt1En)**—Bit 6. Enables MCA reporting of HyperTransport sync error packets detected on HyperTransport link 1. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a sync packet on an incoming link independent of the state of this bit.

**HyperTransport Link 2 Sync Packet Error Reporting Enable (SyncPkt2En)**—Bit 7. Enables MCA reporting of HyperTransport sync error packets detected on HyperTransport link 2. The Northbridge will flood its outgoing HyperTransport links with sync packets after detecting a sync packet on an incoming link independent of the state of this bit.

**Master Abort Error Reporting Enable (MstrAbtEn)**—Bit 8. Enables MCA reporting of master aborts (HyperTransport packets that return an error status with the Non Existent Address bit set). The Northbridge will return an error response back to the requestor with any associated data all 1s independent of the state of this bit.

**Target Abort Error Reporting Enable (TgtAbtEn)**—Bit 9. Enables MCA reporting of target aborts (HyperTransport technology packets that return an error status with the Non Existent Address bit clear). The Northbridge will return an error response back to the requestor with any associated data all 1s independent of the state of this bit.

**GART Table Walk Error Reporting Enable (GartTblWkEn)**—Bit 10. Enables MCA reporting of GART cache table walks which encounter a GART PTE entry which is invalid.

**Atomic Read-Modify-Write Error Reporting Enable (AtomicRMWEn)**—Bit 11. Enables MCA reporting of atomic read-modify-write (RMW) HyperTransport technology commands received from a noncoherent HyperTransport link which do not target DRAM. Atomic RMW commands are not supported by AMD Athlon™ 64 and AMD Opteron™ processors. Atomic RMW commands to memory-mapped I/O are not supported in HyperTransport technology. An atomic RMW command that targets MMIO results in a HyperTransport error response being generated back to the requesting I/O device. The generation of the HyperTransport error response is not affected by this bit.

**Watchdog Timer Error Reporting Enable (WchDogTmrEn)**—Bit 12. Enables MCA reporting of watchdog timer errors. The watchdog timer checks for Northbridge system accesses for which a response is expected and where no response is received. See the MCA NB Configuration register for information regarding configuration of the watchdog timer duration. Note that this bit does not affect operation of the watchdog timer in terms of its ability to complete an access that would otherwise cause a system hang. This bit only affects whether such errors are reported through MCA.

### 3.6.4.2 MCA NB Configuration Register

#### MCA NB Configuration Register

#### Function 3: Offset 44h

31	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1	0
reserved	NbMcaToMstCpuEn	reserved	DisPciCfgCpuErrRsp	IoRdDatErrEn	ChipKillEccEn	EccEn	SyncOnAnyErrEn	SyncOnWdogEn	reserved	GenCrcErrByte1	GenCrcErrByte0	LdtLinkSel	WdogTmrBaseSel	WdogTmrCntSel	WdogTmrDis	IoErrDis	CpuErrDis	IoMstAbortDis	SyncPktPropDis	SyncPktGenDis	SyncOnUcEccEn	CpuRdDatErrEn	CpuEccErrEn					

Bits	Mnemonic	Function	R/W	Reset
31–28	reserved		R	0
27	NbMcaToMstCpuEn	Northbridge MCA to CPU 0 Enable	R/W	0
26	reserved		R	0
25	DisPciCfgCpuErrRsp	PCI configuration CPU Error Response Disable	R/W	0
24	IoRdDatErrEn	I/O Read Data Error Log Enable	R/W	0
23	ChipKillEccEn	Chip-Kill ECC Mode Enable	R/W	0
22	EccEn	ECC Enable	R/W	0
21	SyncOnAnyErrEn	Sync Flood On Any Error Enable	R/W	0
20	SyncOnWdogEn	Sync Flood on Watchdog Timer Error Enable	R/W	0
19–18	reserved		R	0
17	GenCrcErrByte1	Generate CRC Error on Byte Lane 1	R/W	0
16	GenCrcErrByte0	Generate CRC Error on Byte Lane 0	R/W	0
15–14	LdtLinkSel	HyperTransport Link Select for CRC Error Generation	R/W	0
13–12	WdogTmrBaseSel	Watchdog Timer Time Base Select	R/W	0
11–9	WdogTmrCntSel	Watchdog Timer Count Select	R/W	0
8	WdogTmrDis	Watchdog Timer Disable	R/W	0
7	IoErrDis	I/O Error Response Disable	R/W	0
6	CpuErrDis	CPU Error Response Disable	R/W	0
5	IoMstAbortDis	I/O Master Abort Error Response Disable	R/W	0
4	SyncPktPropDis	Sync Packet Propagation Disable	R/W	0
3	SyncPktGenDis	Sync Packet Generation Disable	R/W	0
2	SyncOnUcEccEn	Sync Flood on Uncorrectable ECC Error Enable	R/W	0
1	CpuRdDatErrEn	CPU Read Data Error Log Enable	R/W	0
0	CpuEccErrEn	CPU ECC Error Log Enable	R/W	0

#### Field Descriptions

**CPU ECC Error Log Enable (CpuEccErrEn)**—Bit 0. Enables reporting of ECC errors for data destined for the CPU on this node. This bit should be clear if ECC error logging is enabled for

the remaining error reporting blocks in the CPU. Logging the same error in more than one block may cause a single error event to be treated as a multiple error event and cause the CPU to enter shutdown.

**CPU Read Data Error Log Enable (CpuRdDatErrEn)**—Bit 1. Enables reporting of read data errors (master aborts and target aborts) for data destined for the CPU on this node. This bit should be clear if read data error logging is enabled for the remaining error reporting blocks in the CPU. Logging the same error in more than one block may cause a single error event to be treated as a multiple error event and cause the CPU to enter shutdown.

**Sync Flood on Uncorrectable ECC Error Enable (SyncOnUcEccEn)**—Bit 2. Enables flooding of all HyperTransport links with sync packets on detection of an uncorrectable ECC error.

**Sync Packet Generation Disable (SyncPktGenDis)**—Bit 3. Disables flooding of all outgoing HyperTransport links with sync packets when a CRC error is detected on an incoming link. By default, sync packet generation for CRC errors is controlled through the HyperTransport technology LDTn Link Control registers (see page 56).

**Sync Packet Propagation Disable (SyncPktPropDis)**—Bit 4. Disables flooding of all outgoing HyperTransport links with sync packets when a sync packet is detected on an incoming link. Sync packets are propagated by default.

**I/O Master Abort Error Response Disable (IoMstAbortDis)**—Bit 5. Disables setting the NXA bit in HyperTransport response packets to I/O devices on detection of a master abort HyperTransport technology error condition.

**CPU Error Response Disable (CpuErrDis)**—Bit 6. Disables generation of a read data error response to the CPU core on detection of a target or master abort HyperTransport technology error condition.

**I/O Error Response Disable (IoErrDis)**—Bit 7. Disables setting the Error bit in HyperTransport response packets to I/O devices on detection of a target or master abort HyperTransport technology error condition.

**Watchdog Timer Disable (WdogTmrDis)**—Bit 8. Disables the watchdog timer. The watchdog timer is enabled by default and checks for Northbridge system accesses for which a response is expected and where no response is received. If such a condition is detected the outstanding access is completed by generating an error response back to the requestor. An MCA error may also be generated if enabled in the MCA NB Control register.

**Watchdog Timer Count Select (WdogTmrCntSel)**—Bit 11–9. Selects the count used by the watchdog timer. The counter selected by WdogTmrCntSel determines the maximum count value in the time base selected by WdogTmrBaseSel.

000b = 4095 (default)

001b = 2047

010b = 1023

011b = 511

100b = 255

101b = 127

110b = 63

111b = 31

**Watchdog Timer Time Base Select (WdogTmrBaseSel)**—Bit 13–12. Selects the time base used by the watchdog timer. The counter selected by WdogTmrCntSel determines the maximum count value in the time base selected by WdogTmrBaseSel.

00b = 1 ms (default)

01b = 1  $\mu$ s

10b = 5 ns

11b = reserved

**HyperTransport Link Select for CRC Error Generation (LdtLinkSel)**—Bit 15–14. Selects the HyperTransport link to be used for CRC error injection through GenCrcErrByte1/GenCrcErrByte0.

00b = HyperTransport link 0

01b = HyperTransport link 1

10b = HyperTransport link 2

11b = undefined

**Generate CRC Error on Byte Lane 0 (GenCrcErrByte0)**—Bit 16. Causes a CRC error to be injected on Byte Lane 0 of the HyperTransport link specified by LdtLinkSel. The data carried by the link is unaffected. This bit is cleared after the error has been generated.

**Generate CRC Error on Byte Lane 1 (GenCrcErrByte1)**—Bit 17. Causes a CRC error to be injected on Byte Lane 1 of the HyperTransport link specified by LdtLinkSel. The data carried by the link is unaffected. This bit is cleared after the error has been generated.

**Sync Flood on Watchdog Timer Error Enable (SyncOnWdogEn)**—Bit 20. Enables flooding of all HyperTransport links with sync packets on detection of a watchdog timer error.

**Sync Flood On Any Error Enable (SyncOnAnyErrEn)**—Bit 21. Enables flooding of all HyperTransport links with sync packets on detection of any MCA error that is uncorrectable.

**ECC Enable (EccEn)**—Bit 22. Enables ECC check/correct mode. This bit must be set in order for any ECC checking/correcting to be enabled for any processor block. If set, ECC will be checked and correctable errors will be corrected irrespective of whether machine check ECC reporting is enabled. See “ECC and Chip Kill Error Checking and Correction” on page 137 for more details.

The hardware will only allow values to be programmed into this field which are consistent with the ECC capabilities of the device as specified in the Northbridge Capabilities register (page 165). Attempts to write values inconsistent with the capabilities will result in this field not being updated.

**Chip-Kill ECC Mode Enable (ChipKillEccEn)**—Bit 23. Enables chip-kill ECC mode. Setting this bit causes ECC checking to be based on a 128/16 data/ECC rather than on a 64/8 data/ECC.

Chip-kill ECC can only be enabled in 128-bit DRAM data width mode. It allows correction of an error affecting a single symbol rather than the single bit correction available in 64/8 ECC checking. When this bit is set, EccEn must be set. See “ECC and Chip Kill Error Checking and Correction” on page 137 for more details.

The hardware will only allow values to be programmed into this field which are consistent with the ECC and Chip Kill ECC capabilities of the device as specified in the Northbridge Capabilities register (page 165). Attempts to write values inconsistent with the capabilities will result in this field not being updated.

**I/O Read Data Error Log Enable (IoRdDatErrEn)**—Bit 24. Enables reporting of read data errors (master aborts and target aborts) for data destined for I/O devices. If this bit is clear (default state), master and target aborts for transactions from I/O devices are not logged by MCA, although error responses may still be generated to the requesting I/O device.

**PCI Configuration CPU Error Response Disable (DisPciCfgCpuErrRsp)**—Bit 25. Disables master abort and target abort reporting through the CPU error-reporting banks for PCI configuration accesses. It is recommended that this bit be set in order to avoid MCA exceptions being generated from master aborts for PCI configuration accesses (which can be common during device enumeration).

**Northbridge MCA to CPU 0 Enable (NbMcaToMstExcEn)**—Bit 27. Enables reporting of all MCA errors to CPU 0 including errors from CPU 1. In addition, when this bit is set, reading MC4\_CTL, MC4\_ADDRESS and MC4\_STATUS through MSR accesses from CPU 1 will return zeros and writes that are not all zeros will cause a GP fault. This bit is only valid for dual core processors. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

### 3.6.4.3 MCA NB Status Low Register

#### MCA NB Status Low Register

Function 3: Offset 48h

31	24	23	20	19	16	15	0
Syndrome[15:8]				reserved		ErrorCodeExt	ErrorCode

Bits	Mnemonic	Function	R/W	Reset
31–24	Syndrome[15:8]	Syndrome Bits 15–8 for Chip Kill ECC Mode	R/W	X
23–20	reserved		R	0
19–16	ErrorCodeExt	Extended Error Code	R/W	X
15–0	ErrorCode	Error Code	R/W	X

“X” in the Reset column indicates that the field initializes to an undefined state after reset.

## Field Descriptions

**Error Code (ErrorCode)**—Bits 15–0. Logs an error code when an error is detected. See Table 25 on page 130 for the error codes. Table 18 on page 128 describes the possible error code formats.

**Extended Error Code (ErrorCodeExt)**—Bits 19–16. Logs an extended error code when an error is detected. See Table 25 on page 130 for the extended error codes.

**Syndrome Bits 15–8 for Chip Kill ECC Mode (Syndrome[15–8])**—Bits 31–24. Logs the upper eight syndrome bits when an ECC error is detected. Only valid for ECC errors in Chip-Kill ECC mode.

**Table 18. Error Code Field Formats**

Error Value	Error Type	Description
0000 0000 0001 TTLL	TLB errors	Errors in the GART TLB cache. TT = Transaction Type (Table 19 on page 128) LL = Cache Level (Table 20 on page 128)
0000 0001 RRRR TTLL	Memory errors	Errors in the cache hierarchy (not applicable for Northbridge errors) RRRR = Memory Transaction Type (Table 21 on page 129) TT = Transaction Type (Table 19 on page 128) LL = Cache Level (Table 20 on page 128)
0000 1PPT RRRR IILL	Bus errors	General bus errors including errors in the HyperTransport™ link or DRAM. PP = Participation Processor (Table 22 on page 129) T = Time-out (Table 23 on page 129) RRRR = Memory Transaction Type (Table 21 on page 129) II = Memory or I/O (Table 24 on page 129) LL = Cache Level (Table 20 on page 128)

**Table 19. Transaction Type Bits (TT)**

00b	Instruction
01b	Data
10b	Generic
11b	reserved

**Table 20. Cache Level Bits (LL)**

00b	Level 0 (L0)
01b	Level 1 (L1)
10b	Level 2 (L2)
11b	Generic (LG)



**Table 21. Memory Transaction Type Bits (RRRR)**

0000b	Generic error (GEN)
0001b	Generic read (RD)
0010b	Generic write (WR)
0011b	Data read (DRD)
0100b	Data write (DRW)
0101b	Instruction fetch (IRD)
0110b	Prefetch
0111b	Evict
1000b	Snoop (probe)

**Table 22. Participation Processor Bits (PP)**

00b	Local node originated the request (SRC)
01b	Local node responded to the request (RES)
10b	Local node observed error as 3rd party (OBS)
11b	Generic

**Table 23. Time-Out Bit (T)**

0	Request did not time out
1	Request timed out (TIMOUT)

**Table 24. Memory or I/O Bits (II)**

00b	Memory access (MEM)
01b	reserved
10b	I/O access
11b	Generic (GEN)

Table 25 on page 130 lists the errors reported by the Northbridge, along with the error codes for each error. The error code fields are defined in the table above. See the NB Control register for more information on errors detected by the Northbridge.

**Table 25. Northbridge Error Codes**

Error Type	Extended Error Code	Error Code						
		Type	PP	T	RRRR	II	LL	TT
ECC error	0000	BUS	SRC/RSP	0	RD/WR	MEM	LG	-
CRC error	0001	BUS	OBS	0	GEN	GEN	LG	-
Sync error	0010	BUS	OBS	0	GEN	GEN	LG	-
Mst Abort	0011	BUS	SRC/OBS	0	RD/WR	MEM/IO	LG	-
Tgt Abort	0100	BUS	SRC/OBS	0	RD/WR	MEM/IO	LG	-
GART error	0101	TLB	-	-	-	-	LG	GEN
RMW error	0110	BUS	OBS	0	GEN	IO	LG	-
Wdog error	0111	BUS	GEN	1	GEN	GEN	LG	-
ChipKill ECC error	1000	BUS	SRC/RSP	0	RD/WR	MEM	LG	-

### 3.6.4.4 MCA NB Status High Register

#### MCA NB Status High Register

#### Function 3: Offset 4Ch

31	30	29	28	27	26	25	24	23	22	15	14	13	12	9	8	7	6	4	3	2	1	0	
ErrValid	ErrOver	ErrUnCorr	ErrEn	ErrMiscVal	ErrAddrVal	PCC	reserved	ECC_Synd (7–0)				CorrECC	UnCorrECC	reserved			ErrScrub	reserved	LDTLink		reserved	ErrCPU1	ErrCPU0

Bits	Mnemonic	Function	R/W	Reset
31	ErrValid	Error Valid	R/W	X
30	ErrOver	Error Overflow	R/W	X
29	ErrUnCorr	Error Uncorrected	R/W	X
28	ErrEn	Error Enable	R/W	X
27	ErrMiscVal	Miscellaneous Error Register Valid	R	X
26	ErrAddrVal	Error Address Valid	R/W	X
25	PCC	Processor Context Corrupt	R/W	X
24–23	reserved		R	0
22–15	ECC_Synd (7–0)	Syndrome Bits (7–0) for ECC Errors	R/W	X
14	CorrECC	Correctable ECC Error	R/W	X
13	UnCorrECC	Uncorrectable ECC Error	R/W	X
12–9	reserved		R	0
8	ErrScrub	Error Found by DRAM Scrubber	R/W	X
7	reserved		R	0
6–4	LDTLink	HyperTransport Link Number	R/W	X
3–2	reserved		R	0

Bits	Mnemonic	Function	R/W	Reset
1	ErrCPU1	Error Associated with CPU 1	R/W	X
0	ErrCPU0	Error Associated with CPU 0	R/W	X

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

## Field Descriptions

**Error Associated with CPU 0 (ErrCPU0)**—Bit 0. If set to 1, this bit indicates that the error was associated with CPU0 core.

**Error Associated with CPU 1 (ErrCPU1)**—Bit 1. If set to 1, this bit indicates that the error was associated with CPU1 core.

**HyperTransport Link Number (LDTLink[2:0])**—Bits 6–4. For errors associated with a HyperTransport link (e.g., CRC errors), this field indicates which link was associated with the error.

LDTLink[0] = Error associated with HyperTransport link 0

LDTLink[1] = Error associated with HyperTransport link 1

LDTLink[2] = Error associated with HyperTransport link 2

**Error Found by DRAM Scrubber (ErrScrub)**—Bit 8. If set to 1, this bit indicates that the error was found by the DRAM scrubber.

**Uncorrectable ECC Error (UnCorrECC)**—Bit 13. If set to 1, this bit indicates that the error was an uncorrectable ECC error.

**Correctable ECC Error (CorrECC)**—Bit 14. If set to 1, this bit indicates that the error was a correctable ECC error.

**Syndrome Bits 7–0 for ECC Errors (ECC\_Synd[7:0])**—Bits 22–15. Logs the lower eight syndrome bits when an ECC error is detected.

**Processor Context Corrupt (PCC)**—Bit 25. If set to 1, this bit indicates that the state of the processor may be corrupted by the error condition. Reliable restarting might not be possible.

0 = Processor not corrupted

1 = Processor may be corrupted

**Error Address Valid (ErrAddrVal)**—Bit 26. If set to 1, this bit indicates that the address saved in the address register is the address where the error occurred (i.e. it validates the address in the address register).

0 = Address register not valid

1 = Address register valid

**Miscellaneous Error Register Valid (ErrMiscVal)**—Bit 27. If set to 1, this bit indicates whether the Miscellaneous Error register contains valid information for this error. This bit is read-only 0, since the Miscellaneous Error register is not implemented.

**Error Enable (ErrEn)**—Bit 28. If set to 1, this bit indicates that MCA error reporting is enabled in the MCA Control register.

0 = MCA error reporting not enabled

1 = MCA error reporting enabled

**Error Uncorrected (ErrUnCorr)**—Bit 29. If set to 1, this bit indicates that the error was not corrected by hardware.

0 = Error corrected

1 = Error not corrected

**Error Overflow (ErrOver)**—Bit 30. Set to 1 if the Northbridge detects an error with the valid bit of this register already set. Enabled errors are written over disabled errors, uncorrectable errors are written over correctable errors. Uncorrectable errors are not overwritten. This bit will not be set when an overflow occurs because of a second correctable error in revision D and earlier revisions. However, the error address will be overwritten.

0 = No error overflow

1 = Error overflow

**Error Valid (ErrValid)**—Bit 31. If set to 1, this bit indicates that a valid error has been detected. This bit should be cleared to 0 by software after the register is read.

0 = No valid error detected

1 = Valid error detected

Table 26 lists the errors reported by Northbridge along with the status bits logged for each error. The status bits are defined in the table above. See the Northbridge Control register for more information on errors detected by Northbridge.

**Table 26. Northbridge Error Status Bit Settings**

Error Type	ErrUnCorr	ErrAddr Val	PCC	ECC_Synd Valid	Corr ECC	UnCorr ECC	Err Scrub	LDTLink Valid	Err CPU
ECC error	If multi-bit	1	If multi-bit and src CPU	1	If single-bit	If multi-bit	1/0	0	1/0
CRC error	1	0	1	0	0	0	0	1	0
Sync error	1	0	1	0	0	0	0	1	0
Mst Abort	1	1	If src CPU	0	0	0	0	1	1/0
Tgt Abort	1	1	If src CPU	0	0	0	0	1	1/0
GART error	1	1	If src CPU	0	0	0	0	0	1/0

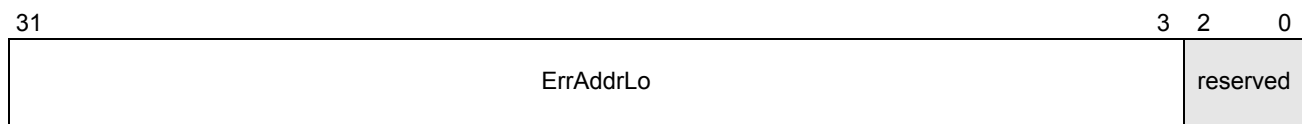
### Table 26. Northbridge Error Status Bit Settings (Continued)

Error Type	ErrUnCorr	ErrAddr Val	PCC	ECC_Synd Valid	Corr ECC	UnCorr ECC	Err Scrub	LDTLink Valid	Err CPU
RMW error	1	1	0	0	0	0	0	1	0
Wdog error	1	0	1	0	0	0	0	0	X
Chip-Kill ECC error	If multi-symbol	1	If multi-symbol and src CPU	1	If single-symbol	If multi-symbol	1/0	0	1/0

### 3.6.4.5 MCA NB Address Low Register

## MCA NB Address Low Register

### Function 3: Offset 50h



Bits	Mnemonic	Function	R/W	Reset
31–3	ErrAddrLo	Error Address Bits 31–3	R/W	X
2–0	reserved		R	0

“X” in the Reset column indicates that the field initializes to an undefined state after reset.

## Field Descriptions

**Error Address Bits 31–3 (ErrAddrLo)**—Bits 31–3. This field specifies bits 31–3 of the address associated with a machine check error.

### 3.6.4.6 MCA NB Address High Register

## MCA NB Address High Register

### Function 3: Offset 54h



Bits	Mnemonic	Function	R/W	Reset
31–8	reserved		R	0
7–0	ErrAddrHi	Error Address Bits 39–32	R/W	X

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

## Field Descriptions

**Error Address Bits 39–32 (ErrAddrHi)**—Bits 7–0. This field specifies bits 39–32 of the address associated with a machine check error.

### 3.6.4.7 Watchdog Timer Errors

For watchdog timer errors the ErrAddrHi and ErrAddrLo fields in the MCA NB Address High/Low registers log the hardware state information of the request for which the response timed out.

#### MCA NB Address Low Register for Watchdog Timer Error Function 3: Offset 50h

Revision B and earlier revisions.

31	30	29	28	27	25	24	23	22	20	19	18	17	15	14	12	11	9	8	3	2	0
WaitLock	WaitPW	WaitCpuDat	WaitDatMove																		reserved

Revision C.

31	30	29	28	27	25	24	23	22	20	19	18	17	15	14	12	11	9	8	3	2	0
WaitCode	WaitPW	reserved																			reserved

Bit	Mnemonic	Function	R/W	Reset
-----	----------	----------	-----	-------

Bits [31:28] in revision B and earlier revisions.

31	WaitLock	Wait For Bus Lock
30	WaitPW	Wait For Posted Write
29	WaitCpuDat	Wait For CPU Write Data
28	WaitDatMove	Wait For Data Movement

Bits [31:28] in revision C.

31-30	WaitCode	Wait Code (Bits 1-0 of WaitCode)
29	WaitPW	Wait For Posted Write
28	reserved	

27–25	DstNode	Destination Node ID
24–23	DstUnit	Destination Unit ID
22–20	SrcNode	Source Node ID
19–18	SrcUnit	Source Unit ID
17–15	SrcPtr	Source Pointer
14–12	NextAction	Next Action
11–9	OpType	Operation Type

Bit	Mnemonic	Function	R/W	Reset
8–3	LdtCmd	HyperTransport Command		
2–0	reserved			

## Field Descriptions

**HyperTransport Command (LdtCmd)**—Bit 8–3. The initial command (in HyperTransport technology format) for the stalled operation.

**Operation Type (OpType)**—Bit 11–9. Indicates what type of operation is stalled.

- 000b = Normal operation (i.e., none of the other types listed)
- 001b = Bus lock
- 010b = Local APIC access
- 011b = Interrupt request
- 100b = System management request
- 101b = Interrupt broadcast
- 110b = Stop grant
- 111b = SMI ack

**Next Action (NextAction)**—Bit 14–12. Indicates what the stalled operation would have done next had it not become stalled.

- 000b = Complete
- 001b = reserved
- 010b = Send request
- 011b = Send second request (if any)
- 100b = Send final response back to requestor
- 101b = Send initial response (if any) back to requestor
- 110b = Send final response to the memory controller
- 111b = Send initial response (if any) to mem controller

**Source Pointer (SrcPtr)**—Bit 17–15. Source pointer of the stalled operation.

- 000b = Host bridge on local node
- 001b = CPU on local node
- 010b = Mem controller on local node
- 101b = HyperTransport link 0
- 110b = HyperTransport link 1
- 111b = HyperTransport link 2

**Source Unit ID (SrcUnit)**—Bit 19–18. Source Unit ID of the stalled operation.

**Source Node ID (SrcNode)**—Bit 22–20. Source Node ID of the stalled operation.

**Destination Unit ID (DstUnit)**—Bit 24–23. Destination Unit ID of the stalled operation.

**Destination Node ID (DstNode)**—Bit 27–25. Destination Node ID of the stalled operation.

Bits [31:28] in revision B and earlier revisions:

**Wait For Data Movement (WaitDatMove)**—Bit 28. When set, indicates that the stalled operation is waiting for read or write data to be moved.

**Wait For CPU Write Data (WaitCpuDat)**—Bit 29. When set, indicates that the stalled operation is waiting for write data from the CPU.

**Wait For Posted Write (WaitPW)**—Bit 30. When set, indicates that a response for the stalled operation is waiting for one or more prior posted writes to complete.

**Wait For Bus Lock (WaitLock)**—Bit 31. When set, indicates that the stalled operation is waiting for a bus lock to complete.

Bits [31:28] in revision C:

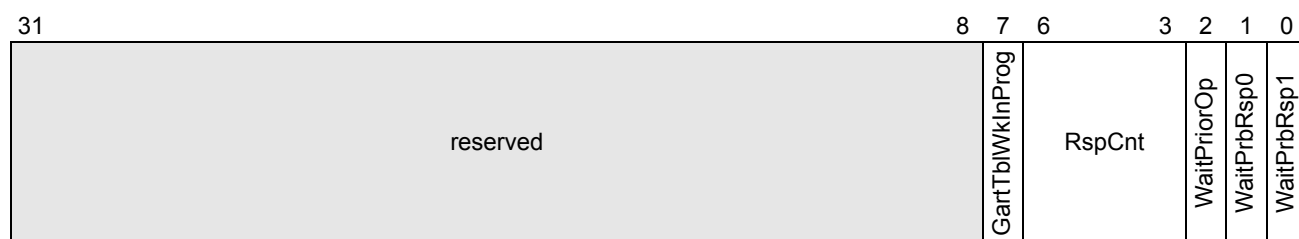
**Wait For Posted Write (WaitPW)**—Bit 29. When set, indicates that a response for the stalled operation is waiting for one or more prior posted writes to complete.

**Wait Code (WaitCode)**—Bit 31-30. WaitCode is a 5 bit field; WaitCode[1:0] is in Function 3, Offset 50h; WaitCode[4:2] is in Function 3, Offset 54h. WaitCode encodings are implementation specific. They describe the reason the hardware is waiting. All zeros mean that there is not a waiting condition. Revision B and earlier revision fields: WaitDatMove, WaitCpuDat, WaitLock (Function 3, Offset 50h), WaitPrbRsp0, WaitPrbRsp1, and WaitPriorOp (Function 3, Offset 50h) are replaced with a code in the revision C field WaitCode (Function 3, Offset 50h, 54h).

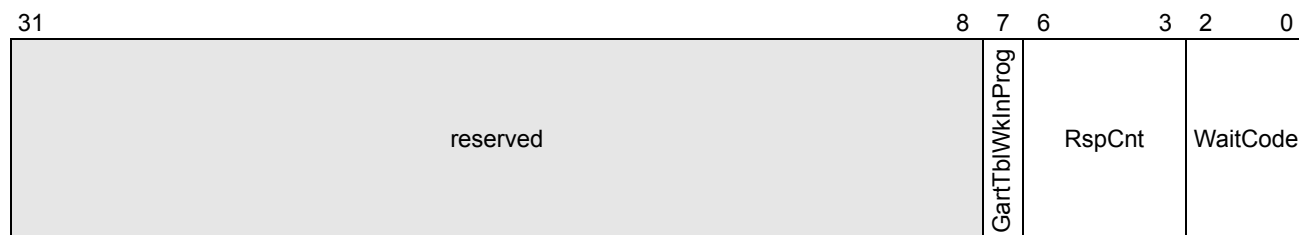
## MCA NB Address High Register for Watchdog Timer Error

## Function 3: Offset 54h

Revision B and earlier revisions.



Revision C.





Bits	Mnemonic	Function	R/W	Reset
31–8	reserved			
7	GartTblWkInProg	GART Table Walk In Progress (Bit 39 of ErrAddr)		
6–3	RspCnt	System Response Count (Bits 38–35 of ErrAddr)		
Bits [2:0] in revision B and earlier revisions.				
2	WaitPriorOp	Wait For Prior Operation (Bit 34 of ErrAddr)		
1	WaitPrbRsp0	Wait For CPU0 Probe Response (Bit 33 of ErrAddr)		
0	WaitPrbRsp1	Wait For CPU1 Probe Response (Bit 32 of ErrAddr)		
Bits [2:0] in revision C.				
2-0	WaitCode	Wait Code (Bits 4-2 of WaitCode)		

## Field Descriptions

Bits [2:0] in revision B and earlier revisions:

**Wait For CPU1 Probe Response (WaitPrbRsp1)**—Bit 0 (Bit 32 of ErrAddr). When set, indicates that the stalled operation is waiting on a probe response from CPU1.

**Wait For CPU0 Probe Response (WaitPrbRsp0)**—Bit 1 (Bit 33 of ErrAddr). When set, indicates that the stalled operation is waiting on a probe response from CPU0.

**Wait For Prior Operation (WaitPriorOp)**—Bit 2 (Bit 34 of ErrAddr). When set, indicates that the stalled operation is waiting for a prior operation to release system resources.

Bits [2:0] in revision C:

**Wait Code (WaitCode)**—Bit 2-0. See WaitCode definition in MCA NB Address Low Register for Watchdog Timer Error (Function 3, Offset 50h).

**System Response Count (RspCnt)**—Bits 6–3 (Bits 38–35 of ErrAddr). Number of outstanding system responses for which the stalled operation is waiting.

**GART Table Walk In Progress (GartTblWkInProg)**—Bit 7 (Bit 39 of ErrAddr). Indicates that a GART table walk was in progress at the time the error was logged.

## 3.6.5 ECC and Chip Kill Error Checking and Correction

The memory controller implements two ECC modes: normal ECC and Chip Kill ECC. These error checking and correction modes can only be used if all DIMMs are ECC capable.

### 3.6.5.1 Normal ECC

Normal ECC mode is a 64/8 SEC/DED (Single Error Correction/Double Error Detection) Hamming code. It can detect one bit error and correct it, it can detect two bit errors but it can not correct them, and it may detect more than two bit errors depending on the position of corrupted bits. Normal ECC mode is enabled when EccEn, Function 3, Offset 44h is set and ChipKillEccEn, Function 3, Offset 44h is clear.

Error address (Function 3, Offsets 50h and 54h) is valid and uniquely identifies the DIMM with a correctable or uncorrectable error for 64-bit and 128-bit data width configurations. In a 128-bit data width configuration, ECC is independently applied to the lower 64 and upper 64 data bits. In this configuration ECC may detect and correct two bit errors if one error happens on data bits 63-0 and one error happens on data bits 127-64. Status registers (Function 3, Offsets 48h and 4Ch) and error address registers will contain information about the second detected error.

Bit position of a correctable error can be determined by matching Ecc\_Synd, Function 3, Offset 4Ch with a value from Table 27. Bits 63-0 correspond to data bits, and bits 73-64 correspond to ECC check bits.

**Table 27. ECC Syndromes**

	n = 0	n = 1	n = 2	n = 3	n = 4	n = 5	n = 6	n = 7
Bit (0+n)	ce	cb	d3	d5	d6	d9	da	dc
Bit (8+n)	23	25	26	29	2a	2c	31	34
Bit (16+n)	0e	0b	13	15	16	19	1a	1c
Bit (24+n)	e3	e5	e6	e9	ea	ec	f1	f4
Bit (32+n)	4f	4a	52	54	57	58	5b	5d
Bit (40+n)	a2	a4	a7	a8	ab	ad	b0	b5
Bit (48+n)	8f	8a	92	94	97	98	9b	9d
Bit (56+n)	62	64	67	68	6b	6d	70	75
Bit (64+n)	01	02	04	08	10	20	40	80

### 3.6.5.2 Chip Kill

Chip Kill ECC mode is a 128/16 SSC/DSD (Single Symbol Correction/Double Symbol Detection) BCH code. A symbol is a group of 4 bits which are 4 bit aligned (bits 0–3 make symbol 0, bits 4–7 make symbol 1, etc.). A single symbol error is any bit error combination within one symbol (1 to 4 bits can be corrupted). Chip Kill ECC mode can detect one symbol error and correct it, it can detect two symbol errors but it can not correct them, and it may detect more than two symbol errors depending on the position of corrupted symbols. Chip Kill ECC mode is enabled when EccEn and ChipKillEccEn, Function 3, Offset 44h are set.

Chip Kill mode can only be used in a 128-bit data width configuration.

A DIMM with a correctable error is uniquely identified with error address (Function 3, Offsets 50h and 54h) and Chip Kill syndrome (Syndrome, Function 3, Offset 48h and Ecc\_Synd, Function 3, Offset 4Ch). Error address identifies two DIMMs and Chip Kill syndrome identifies one of them. Chip Kill syndromes for symbols 00h-0fh map to data bits 0-63; Chip Kill syndromes for symbols 20-21h map to ECC check bits for data bits 0-63; Chip Kill syndromes for symbols 10h-1fh map to data bits 64-127; Chip Kill syndromes for symbols 22-23h map to ECC check bits for data bits 64-127 (see Table 28). Corrupted bit positions within a symbol are determined from the Chip Kill syndrome

column number in Table 28. Bits set in the column number identify corrupted bits within a symbol. For example, if 6913h is a Chip Kill syndrome, symbol 05h has an error, and bits 0 and 1 within that symbol are corrupted, since the syndrome is in column 3h. Symbol 05h maps to bits 23-20, so the corrupted bits are 20 and 21.

A DIMM with an uncorrectable error can not be uniquely identified. The error address is valid and maps to two DIMMs.

Chip Kill mode can be used with any device width. Chip Kill mode can correct all errors in an x4 device, since the device width is one symbol. Chip Kill mode can correct a subset of errors in an x8 or x16 device.

**Table 28. Chip Kill ECC Syndromes**

Symbol	1h	2h	3h	4h	5h	6h	7h	8h	9h	ah	bh	ch	dh	eh	fh
00h	e821	7c32	9413	bb44	5365	c776	2f57	dd88	35a9	a1ba	499b	66cc	8eed	1afe	f2df
01h	5d31	a612	fb23	9584	c8b5	3396	6ea7	eac8	b7f9	4cda	11eb	7f4c	227d	d95e	846f
02h	0001	0002	0003	0004	0005	0006	0007	0008	0009	000a	000b	000c	000d	000e	000f
03h	2021	3032	1013	4044	6065	7076	5057	8088	a0a9	b0ba	909b	c0cc	e0ed	f0fe	d0df
04h	5041	a082	f0c3	9054	c015	30d6	6097	e0a8	b0e9	402a	106b	70fc	20bd	d07e	803f
05h	be21	d732	6913	2144	9f65	f676	4857	3288	8ca9	e5ba	5b9b	13cc	aded	c4fe	7adf
06h	4951	8ea2	c7f3	5394	1ac5	dd36	9467	a1e8	e8b9	2f4a	661b	f27c	bb2d	7cde	358f
07h	74e1	9872	ec93	d6b4	a255	4ec6	3a27	6bd8	1f39	f3aa	874b	bd6c	c98d	251e	51ff
08h	15c1	2a42	3f83	cef4	db35	e4b6	f177	4758	5299	6d1a	78db	89ac	9c6d	a3ee	b62f
09h	3d01	1602	2b03	8504	b805	9306	ae07	ca08	f709	dc0a	e10b	4f0c	720d	590e	640f
0ah	9801	ec02	7403	6b04	f305	8706	1f07	bd08	2509	510a	c90b	d60c	4e0d	3a0e	a20f
0bh	d131	6212	b323	3884	e9b5	5a96	8ba7	1cc8	cdf9	7eda	afeb	244c	f57d	465e	976f
0ch	e1d1	7262	93b3	b834	59e5	ca56	2b87	dc18	3dc9	ae7a	4fab	642c	85fd	164e	f79f
0dh	6051	b0a2	d0f3	1094	70c5	a036	c067	20e8	40b9	904a	f01b	307c	502d	80de	e08f
0eh	a4c1	f842	5c83	e6f4	4235	1eb6	ba77	7b58	df99	831a	27db	9dac	396d	65ee	c12f
0fh	11c1	2242	3383	c8f4	d935	eab6	fb77	4c58	5d99	6e1a	7fdb	84ac	956d	a6ee	b72f
10h	45d1	8a62	cfb3	5e34	1be5	d456	9187	a718	e2c9	2d7a	68ab	f92c	bcfd	734e	369f
11h	63e1	b172	d293	14b4	7755	a5c6	c627	28d8	4b39	99aa	fa4b	3c6c	5f8d	8d1e	eeff
12h	b741	d982	6ec3	2254	9515	fbdc	4c97	33a8	84e9	ea2a	5d6b	11fc	a6bd	c87e	7f3f
13h	dd41	6682	bbc3	3554	e815	53d6	8e97	1aa8	c7e9	7c2a	a16b	2ffc	f2bd	497e	943f
14h	2bd1	3d62	16b3	4f34	64e5	7256	5987	8518	aec9	b87a	93ab	ca2c	e1fd	f74e	dc9f
15h	83c1	c142	4283	a4f4	2735	65b6	e677	f858	7b99	391a	badb	5cac	df6d	9dee	1e2f
16h	8fd1	c562	4ab3	a934	26e5	6c56	e387	fe18	71c9	3b7a	b4ab	572c	d8fd	924e	1d9f
17h	4791	89e2	ce73	5264	15f5	db86	9c17	a3b8	e429	2a5a	6dcb	f1dc	b64d	783e	3faf

**Table 28. Chip Kill ECC Syndromes**

Symbol	1h	2h	3h	4h	5h	6h	7h	8h	9h	ah	bh	ch	dh	eh	fh
18h	5781	a9c2	fe43	92a4	c525	3b66	6ce7	e3f8	b479	4a3a	1dbb	715c	26dd	d89e	8f1f
19h	bf41	d582	6ac3	2954	9615	fcd6	4397	3ea8	81e9	eb2a	546b	17fc	a8bd	c27e	7d3f
1ah	9391	e1e2	7273	6464	f7f5	8586	1617	b8b8	2b29	595a	cacb	dcdc	4f4d	3d3e	aeaf
1bh	cce1	4472	8893	fdb4	3155	b9c6	7527	56d8	9a39	12aa	de4b	ab6c	678d	ef1e	23ff
1ch	a761	f9b2	5ed3	e214	4575	1ba6	bcc7	7328	d449	8a9a	2dfb	913c	365d	688e	cfef
1dh	ff61	55b2	aad3	7914	8675	2ca6	d3c7	9e28	6149	cb9a	34fb	e73c	185d	b28e	4def
1eh	5451	a8a2	fcf3	9694	c2c5	3e36	6a67	ebe8	bfb9	434a	171b	7d7c	292d	d5de	818f
1fh	6fc1	b542	da83	19f4	7635	acb6	c377	2e58	4199	9b1a	f4db	37ac	586d	82ee	ed2f
20h	be01	d702	6903	2104	9f05	f606	4807	3208	8c09	e50a	5b0b	130c	ad0d	c40e	7a0f
21h	4101	8202	c303	5804	1905	da06	9b07	ac08	ed09	2e0a	6f0b	f40c	b50d	760e	370f
22h	c441	4882	8cc3	f654	3215	bed6	7a97	5ba8	9fe9	132a	d76b	adfc	69bd	e57e	213f
23h	7621	9b32	ed13	da44	ac65	4176	3757	6f88	19a9	f4ba	829b	b5cc	c3ed	2efe	58df

### 3.6.6 Scrub Control Register

This register specifies the scrub rate for sequential ECC scrubbing of DRAM, the L2 cache and the DCACHE. The scrub rate specifies the duration between successive scrub events. A value of 0 disables scrubbing.

Each scrub event corresponds to:

- 64 bytes for the DRAM scrubber.
- 64 bits for the data cache scrubber.
- One single L2 cache line tag address for the L2 scrubber.

#### Scrub Control Register

#### Function 3: Offset 58h

31	21	20	16	15	13	12	8	7	5	4	0
reserved				DcacheScrub		reserved	L2Scrub		reserved	DramScrub	

Bits	Mnemonic	Function	R/W	Reset
31–21	reserved		R	0
20–16	DcacheScrub	Data Cache Scrub Rate	R/W	0
15–13	reserved		R	0
12–8	L2Scrub	L2 Cache Scrub Rate	R/W	0
7–5	reserved		R	0
4–0	DramScrub	DRAM Scrub Rate	R/W	0

## Field Descriptions

**DRAM Scrub Rate (DramScrub)**—Bits 4–0. Specifies the scrub rate for the DRAM. See Table 29.

For example, if 256MByte memory is scrubbed every 12 hours, a 64-byte memory block should be scrubbed every 10ms, and scrubbing rate of 10.49ms should be selected.

**L2 Cache Scrub Rate (L2Scrub)**—Bits 12–8. Specifies the scrub rate for the L2 cache. See Table 29.

**Data Cache Scrub Rate (DcacheScrub)**—Bits 20–16. Specifies the scrub rate for the data cache. See Table 29.

**Table 29. Scrub Rate Control Values**

Scrub Rate Code	Scrub Rate	Scrub Rate Code	Scrub Rate
00000b	Do not scrub	01100b	81.9 $\mu$ s
00001b	40.0 ns	01101b	163.8 $\mu$ s
00010b	80.0 ns	01110b	327.7 $\mu$ s
00011b	160.0 ns	01111b	655.4 $\mu$ s
00100b	320.0 ns	10000b	1.31 ms
00101b	640.0 ns	10001b	2.62 ms
00110b	1.28 $\mu$ s	10010b	5.24 ms
00111b	2.56 $\mu$ s	10011b	10.49 ms
01000b	5.12 $\mu$ s	10100b	20.97 ms
01001b	10.2 $\mu$ s	10101b	42.00 ms
01010b	20.5 $\mu$ s	10110b	84.00 ms
01011b	41.0 $\mu$ s	All others	reserved

### 3.6.7 DRAM Scrub Address Registers

These registers specify the next address to be scrubbed by the DRAM scrubber. They should be initialized by the BIOS before the scrubber is enabled (the DRAM scrubber is enabled by writing a valid scrub rate to DramScrub field, Function 3, Offset 58h) and after MemClrStatus, Function 2, Offset 90h is set. They are then updated by the scrubber hardware as it scrubs successive 64-byte blocks of DRAM. Once the scrubber reaches the DRAM limit address for the node (see “DRAM Address Map” on page 70), it wraps to the DRAM base address.

The initial scrubbing address specified by these registers must be between the base and limit address for the node, defined through the DRAM address maps (see “DRAM Address Map” on page 70). The recommended initial scrubbing address is the DRAM base address.

In addition to sequential DRAM scrubbing, the DRAM scrubber has a redirect mode for scrubbing DRAM locations accessed during normal operation. This is enabled by setting ScrubReDirEn (Function 3, Offset 5Ch). When a DRAM read is generated by any agent other than the DRAM

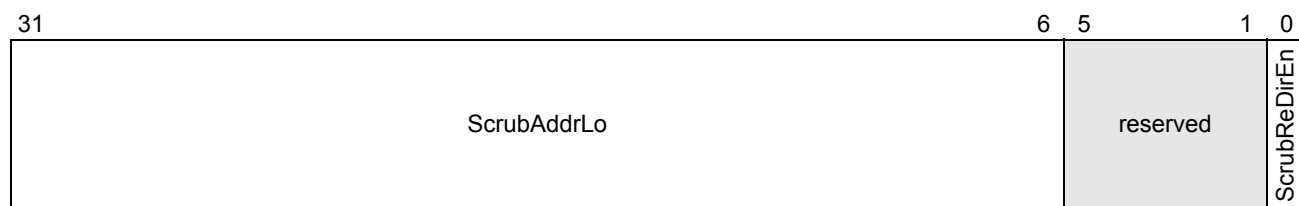
scrubber, correctable ECC errors are corrected as the data is passed to the requestor, but the data in DRAM is not corrected if redirect scrubbing mode is disabled. In scrubber redirect mode, correctable errors detected during normal DRAM read accesses redirect the scrubber to the location of the error. After the scrubber corrects the location in DRAM, it resumes scrubbing from where it left off. DRAM scrub address registers are not modified by the redirect scrubbing mode. Sequential scrubbing and scrubber redirection can be enabled independently or together.

ECC errors detected by the scrubber are logged in the MCA registers (see “Machine Check Architecture Registers” on page 192).

### 3.6.7.1 DRAM Scrub Address Low Register

#### DRAM Scrub Address Low Register

Function 3: Offset 5Ch



Bits	Mnemonic	Function	R/W	Reset
31–6	ScrubAddrLo	DRAM Scrub Address Bits 31–6	R/W	X
5–1	reserved		R	0
0	ScrubReDirEn	DRAM Scrubber Redirect Enable	R/W	0

“X” in the Reset column indicates that the field initializes to an undefined state after reset.

#### Field Descriptions

**DRAM Scrubber Redirect Enable (ScrubReDirEn)**—Bit 0. If this bit is set, the scrubber is redirected to correct errors found during normal operation.

**DRAM Scrub Address Bits 31–6 (ScrubAddrLo)**—Bits 31–6. This field specifies the low address bits 31–6 of the next 64-byte block to be scrubbed.

### 3.6.7.2 DRAM Scrub Address High Register

#### DRAM Scrub Address High Register

Function 3: Offset 60h



Bits	Mnemonic	Function	R/W	Reset
31–8	reserved		R	0
7–0	ScrubAddrHi	DRAM Scrub Address Bits 39–32	R/W	X

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

## Field Descriptions

**DRAM Scrub Address Bits 39–32 (ScrubAddrHi)**—Bits 7–0. This field specifies the high address bits 39–32 of the next 64-byte block to be scrubbed.

### 3.6.8 XBAR Flow Control Buffers

The Northbridge interfaces with the CPU core, DRAM controller, and, through three HyperTransport links, to external chips.

The major Northbridge blocks are: System Request Interface (SRI), Memory Controller (MCT), and Cross Bar (XBAR). SRI interfaces with the CPU core and connects coherent HyperTransport links and noncoherent HyperTransport links. MCT maintains cache coherency and interfaces with the DRAM. XBAR is a five port switch which routes the command packets between SRI, MCT, and the three HyperTransport links. Not all HyperTransport links have to be active.

The number of buffers available for each link at the XBAR input is shown in Table 30.

**Table 30. XBAR Input Buffers**

Link	Number of Command Buffers	Number of Data Buffers
HyperTransport™ Link 0-2	16 x 3	8 x 3
SRI	10	5
MCT	12	8
<b>Total</b>	70	37

XBAR command and data buffers are independent. There are 70 command buffers available, but a maximum of 64 can be used at any given time. Number of used data buffers is not restricted. The default allocation of command buffers when all HyperTransport links are present is shown in Table 31.

**Table 31. Default XBAR Command Buffer Allocation**

Link	Number of Command Buffers
HyperTransport™ Link 0-2	15 x 3
SRI	8

**Table 31. Default XBAR Command Buffer Allocation**

Link	Number of Command Buffers
MCT	11
<b>Total</b>	<b>64</b>

The HyperTransport™ I/O Link Specification defines four virtual channels: requests, posted requests, responses, and probes. The default virtual channel command buffer allocation is shown in Table 32. At least one command buffer should be allocated to each used virtual channel to avoid deadlock. Command buffers do not need to be allocated for a virtual channel that is not used by a link. For example, MCT does not initiate any requests, so there is no need to allocate request or posted request command buffers for an MCT link. A link should not send transactions through a virtual channel that does not have at least one command buffer allocated.

Default allocation of SRI buffers can be found in “SRI-to-XBAR Buffer Count Register” on page 147 and “Free List Buffer Count Register” on page 149. In the SRI-to-Xbar Buffer Count Register (Function 3, Offset 70h), the virtual channels are subdivided into upstream (coherent HyperTransport) and downstream (noncoherent HyperTransport) directions, since SRI must send packets into both coherent HyperTransport and noncoherent HyperTransport links. Some buffers are allocated to a free list pool to use the available resources more efficiently. The buffers in the free list pool are shared by packets in multiple virtual channels. By default, no more than one buffer is allocated to a virtual channel and the rest are allocated to the free list pool in the Free List Buffer Count Register (Function 3, Offset 7Ch). Any buffer increase to SRI can be added to the free list or allocated directly to a virtual channel. The total count allocated through the Free List Buffer Count Register and the SRI-to-Xbar Buffer Count Register cannot exceed 10.

**Table 32. Default Virtual Channel Command Buffer Allocation**

Link	Request	Posted Request	Response	Probe	Number of Command Buffers
Coherent HyperTransport™ Links	3	1	6	5	15
Non-coherent HyperTransport™ Links	6	5	4	0	15
SRI	2	3	3	0	8
MCT	0	0	8	3	11

Software can reallocate buffers by modifying buffer count registers (Function 0, Offsets 90h, B0h, D0h, Function 3, Offsets 70h, 78h, 7Ch). The new buffer counts take effect after a warm reset. Since hardware attempts to choose optimal settings, in general, these registers should not be changed.

The following information should be used for buffer reallocation:

1. Number of allocated command buffers for each link should not be greater than the number of available command buffers for that link (see Table 30). Sum of allocated command buffers for three HyperTransport links, SRI, and MCT should not be greater than 64.



2. If one HyperTransport link is not present, then other links can use all available buffers for each of them (MCT and HyperTransport links can use one additional buffer, and SRI can use two additional buffers).
3. If all links are present, some buffers from a noncoherent HyperTransport link can be reallocated to coherent HyperTransport links. Table 33 shows command buffer allocation after two response buffers from noncoherent HyperTransport link 0 are reallocated to coherent HyperTransport links 1 and 2.

**Table 33. An Example of a Non Default Virtual Channel Command Buffer Allocation**

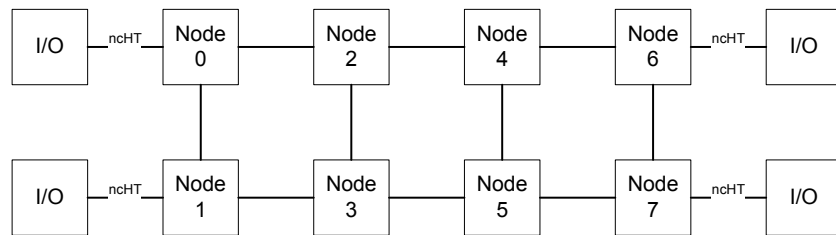
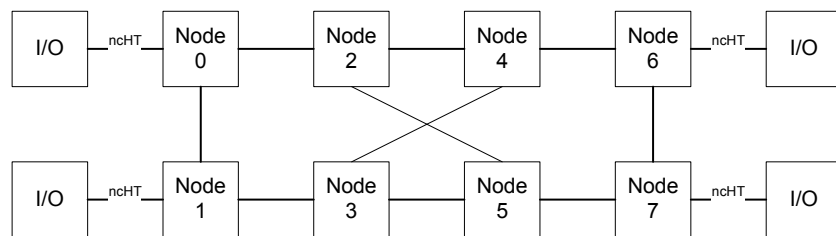
Link	Request	Posted Request	Response	Probe	Number of Command Buffers
HyperTransport™ Link 0	6	5	2	0	13
HyperTransport™ Link 1	3	1	7	5	16
HyperTransport™ Link 2	3	1	7	5	16
SRI	2	3	3	0	8
MCT	0	0	8	3	11

4. If there are no noncoherent HyperTransport links, one or more response buffers can be reallocated from MCT to HyperTransport links.
5. In a multiprocessor system, number of coherent HyperTransport response buffers should be increased first. The default buffer allocation is sufficient for a uniprocessor system.
6. If a two processor system is implemented with two coherent HyperTransport links between the processors, AMD recommends configuring one link for requests and the other link for responses using the buffer allocations in Table 34.

**Table 34. 2P Dual Coherent HyperTransport Command Buffer Allocation**

Link	Request	Posted Request	Response	Probe	Number of Command Buffers
Coherent HyperTransport™ Request Link	6	1	3	6	15
Coherent HyperTransport™ Response Link	0	0	15	0	16
Non-coherent HyperTransport™ Links	6	5	2	0	13
SRI	2	3	3	0	8
MCT	0	0	8	3	11

7. AMD recommends using different buffer allocations for inner and outer nodes when an eight processor system is implemented using either a ladder (Figure 2) or a twisted ladder (Figure 3) configuration. An outer node is a node with a noncoherent HyperTransport link. Unconnected links are never considered noncoherent HyperTransport links when determining if the node is an inner or outer node. Table 35 and Table 36 show the recommended buffer allocations.


**Figure 2. 8P Ladder Configuration**

**Figure 3. 8P Twisted Ladder Configuration**
**Table 35. 8P Outer Node Virtual Channel Command Buffer Allocation**

Link	Request	Posted Request	Response	Probe	Number of Command Buffers
Coherent HyperTransport™ Links	1	1	8	5	15
Non-coherent HyperTransport™ Links	6	5	4	0	15
SRI	2	3	3	0	8
MCT	0	0	8	3	11

**Table 36. 8P Inner Node Virtual Channel Command Buffer Allocation**

Link	Request	Posted Request	Response	Probe	Number of Command Buffers
Coherent HyperTransport™ Links	3	1	6	5	15
SRI	2	3	3	0	8
MCT	0	0	8	3	11

8. AMD recommends that UMA graphics systems use the flow control buffer allocation in Table 37. Note that some chipsets may require different values from those specified here. Contact the chipset vendor for specific recommendations.

**Table 37. Recommended Flow Control Buffer Allocations in UMA systems**

Dev:0x90 LDT		Dev:3x70 SRI-XBAR		Dev:3x74 XBAR-SRI		Dev:3x78 MCT-XBAR		Dev:3x7C Free List	
RspD	1	DPreq	1	DPreq	1	RspD	8	FRspD	2
PReqD	5	DReq	1	DReq	1	Prb	2	FRsp	1
ReqD	2	URspD	1	DispRefReq	7	Rsp	10	FReq	1
Probe	0	DispRefReq	3	Prb	4			FreeCmd	8 <sup>1/72</sup>
Rsp	1	ReqD	2	UPReq	2				
PReq	5	URsp	1	UReq	1				
Req	10	UPReq	1						
		UReq	1						
1. Single Core Processor 2. Dual Core Processor									

### 3.6.8.1 SRI-to-XBAR Buffer Count Register

#### SRI-to-XBAR Buffer Count Register

#### Function 3: Offset 70h

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15		10	9	8	7	6	5	4	3	2	1	0	
DPReq		DReq		reserved		URspD		reserved		DispRefReq		reserved		ReqD		reserved			URsp			reserved		UPReq		reserved		UReq	

Bits	Mnemonic	Function	R/W	Reset
31–30	DPreq	Downstream Posted Request Buffer Count	R/W	01b
29–28	DReq	Downstream Request Buffer Count	R/W	01b
27–26	reserved		R	0
25–24	URspD	Upstream Response Data Buffer Count	R/W	01b
23–22	reserved		R	0
21:20	DispRefReq	Display Refresh Request Buffer Count	R/W	0
19–18	reserved		R	0
17–16	ReqD	Request Data Buffer Count	R/W	10b
15–10	reserved		R	0
9–8	URsp	Upstream Response Buffer Count	R/W	01b
7–6	reserved		R	0
5–4	UPReq	Upstream Posted Request Buffer Count	R/W	01b
3–2	reserved		R	0
1–0	UReq	Upstream Request Buffer Count	R/W	01b

## Field Descriptions

**Upstream Request Buffer Count (UReq)**—Bits 1–0. This field defines the number of upstream request buffers available in the XBAR for use by the SRI.

**Upstream Posted Request Buffer Count (UPReq)**—Bits 5–4. This field defines the number of upstream posted request buffers available in the XBAR for use by the SRI.

**Upstream Response Buffer Count (URsp)**—Bits 9–8. This field defines the number of upstream response buffers available in the XBAR for use by the SRI.

**Request Data Buffer Count (ReqD)**—Bits 17–16. This field defines the number of request data buffers available in the XBAR for use by the SRI.

**Display Refresh Request Buffer Count (DispRefReq)**—Bits 21–20. This field defines the number of display refresh request buffers available in the XBAR for use by the SRI. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Upstream Response Data Buffer Count (URspD)**—Bits 25–24. This field defines the number of upstream response data buffers available in the XBAR for use by the SRI.

**Downstream Request Buffer Count (DReq)**—Bits 29–28. This field defines the number of downstream request buffers available in the XBAR for use by the SRI.

**Downstream Posted Request Buffer Count (DPReq)**—Bits 31–30. This field defines the number of downstream posted request buffers available in the XBAR for use by the SRI.

### 3.6.8.2 MCT-to-XBAR Buffer Count Register

#### MCT-to-XBAR Buffer Count Register

Function 3: Offset 78h

31	28	27	24	23	15	14	12	11	8	7	0
reserved		RspD		reserved		Prb		Rsp		reserved	

Bits	Mnemonic	Function	R/W	Reset
31–28	reserved		R	0
27–24	RspD	Response Data Buffer Count	R/W	8h
23–15	reserved		R	0
14–12	Prb	Probe Buffer Count	R/W	011b
11–8	Rsp	Response Buffer Count	R/W	8h
7–0	reserved		R	0

## Field Descriptions

**Response Buffer Count (Rsp)**—Bits 11–8. This field defines the number of response buffers available in the XBAR for use by the MCT.

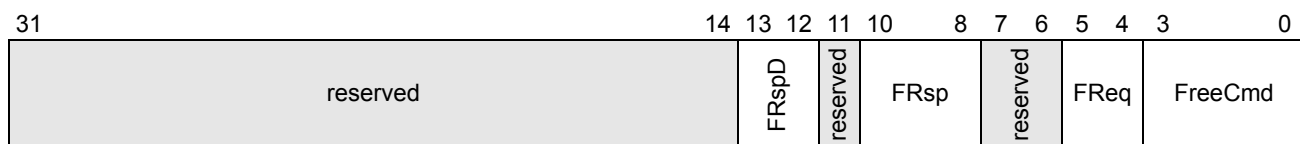
**Probe Buffer Count (Prb)**—Bits 14–12. This field defines the number of probe buffers available in the XBAR for use by the MCT.

**Response Data Buffer Count (RspD)**—Bits 27–24. This field defines the number of response data buffers available in the XBAR for use by the MCT.

### 3.6.8.3 Free List Buffer Count Register

## Free List Buffer Count Register

### Function 3: Offset 7Ch



Bits	Mnemonic	Function	R/W	Reset
31–14	reserved		R	0
13–12	FRspD	SRI to XBAR Free Response Data Buffer Count	R/W	10b
11	reserved		R	0
10–8	FRsp	SRI to XBAR Free Response Buffer Count	R/W	010b
7–6	reserved		R	0
5–4	FReq	SRI to XBAR Free Request Buffer Count	R/W	01b
3–0	FreeCmd	SRI Free Command Buffer Count	R/W	Bh <sup>1</sup> /Ah <sup>2</sup>

1. Single Core Processor
2. Dual Core Processor

## Field Descriptions

**SRI Free Command Buffer Count (FreeCmd)**—Bits 3–0. This field defines the number of free list request or posted request buffers available in the SRI for use by the XBAR or CPU.

**SRI to XBAR Free Request Buffer Count (FReq)**—Bits 5–4. This field defines the number of free list request buffers available in the XBAR for use by the SRI.

**SRI to XBAR Free Response Buffer Count (FRsp)**—Bits 10–8. This field defines the number of free list response buffers available in the XBAR for use by the SRI.

**SRI to XBAR Free Response Data Buffer Count (FRspD)**—Bits 13–12. This field defines the number of free list response data buffers available in the XBAR for use by the SRI.

### 3.6.9 XBAR-to-SRI Buffer Count Register

The Cross Bar Switch to System Request Interface (XBAR-to-SRI) Buffer Count register specifies the number of command buffers for each virtual channel available in the SRI for use by the XBAR. These buffers store commands for traffic routed to the SRI by the XBAR. The buffer counts take effect on the next warm reset.

Since the XBAR must route packets in both the coherent HyperTransport technology fabric and down the noncoherent HyperTransport chains, the usual virtual channels are further subdivided into upstream (from noncoherent HyperTransport) and downstream (to noncoherent HyperTransport) directions. In order to make more efficient use of the available resources, some of the buffers are allocated onto a free list pool in which case these buffers can be used by packets issued in either direction. See “Free List Buffer Count Register” on page 149. Any increase in the number of fixed allocated buffers in the XBAR-to-SRI Buffer Count register must be compensated by a corresponding reduction of the number of free list buffers specified in the Free List Buffer Count register.

When modifying buffer counts care must be taken to allocate a minimum number of buffers to each virtual channel to avoid deadlock. Requests and Posted requests in both directions require at least one buffer and Probes require at least two buffers to avoid deadlock. Since hardware attempts to choose optimal settings, this register should not in general need to be changed.

**Note:** When the Probe Buffer Count is changed, care must be taken to avoid generation of system management events from the time the new value is written into this register until it takes effect (on the next warm reset).

#### XBAR-to-SRI Buffer Count Register

#### Function 3: Offset 74h

31	30	29	28	27	23	22	20	19	16	15	12	11	7	6	4	3	2	0
DPreq	DReq	reserved				DispRefReq	reserved				Prb	reserved				UPReq	reserved	UReq

Bits	Mnemonic	Function	R/W	Reset
31–30	DPreq	Downstream Posted Request Buffer Count	R/W	01b
29–28	DReq	Downstream Request Buffer Count	R/W	01b
27–23	reserved		R	0
22:20	DispRefReq	Display Refresh Request Buffer Count	R/W	0
19–16	reserved		R	0
15–12	Prb	Probe Buffer Count	R/W	8h
11–7	reserved		R	0
6–4	UPReq	Upstream Posted Request Buffer Count	R/W	001b
3	reserved		R	0
2–0	UReq	Upstream Request Buffer Count	R/W	001b

## Field Descriptions

**Upstream Request Buffer Count (UReq)**—Bits 2–0. This field defines the number of upstream request buffers available in the SRI for use by the XBAR.

**Upstream Posted Request Buffer Count (UPReq)**—Bits 6–4. This field defines the number of upstream posted request buffers available in the SRI for use by the XBAR.

**Probe Buffer Count (Prb)**—Bits 15–12. This field defines the number of probe buffers available in the SRI for use by the XBAR.

**Display Refresh Request Buffer Count (DispRefReq)**—Bits 22–20. This field defines the number of display refresh request buffers available in the SRI for use by the XBAR. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Downstream Request Buffer Count (DReq)**—Bits 29–28. This field defines the number of downstream request buffers available in the SRI for use by the XBAR.

**Downstream Posted Request Buffer Count (DPReq)**—Bits 31–30. This field defines the number of downstream posted request buffers available in the SRI for use by the XBAR.

### 3.6.10 Display Refresh Flow Control Buffers

The Northbridge has a separate logical path from HyperTransport links to the system memory for display refresh requests. This is necessary to support the bandwidth and latency requirements of display refresh requests in systems where the display-refresh frame buffer is located in the system memory.

A HyperTransport packet is defined as a display-refresh request packet when the read request has isochronous bit, PassPW bit, and RspPassPW bit set, and coherent bit and SeqID cleared. All display-refresh requests are marked as high priority requests and will have higher priority than other requests. In order to accomplish a dedicated logical path to system memory a minimum number of buffers have to be allocated in various queues.

DispRefReq (Function 3, Offset 70h) and DispRefReq (Function 3, Offset 74h) need to be set to allocate the buffers for display refresh requests. At least one buffer should be allocated for display refresh requests to avoid deadlock. Usually more than one buffer may be required to support the necessary bandwidth. The buffer count written to the registers takes effect after a warm reset. See “SRI-to-XBAR Buffer Count Register” on page 147 and “XBAR-to-SRI Buffer Count Register” on page 150 for more information on setting these registers.

### 3.6.11 Power Management Control Registers

These registers specify the processor response to STPCLK and HALT power management events. Each STPCLK request received contains a 3-bit System Management Action Field (SMAF) which is used as an index into the Power Management Control Low/High registers to select a Power

Management Mode (PMM) value to use for the duration of the STPCLK event. A SMAF value of 000 selects PMM0, a SMAF value of 001 selects PMM1, and so on. HALT is hardwired to use PMM7.

### 3.6.11.1 Power Management Mode (PMM) Value

The PMM value contains the following fields:

Bit	Name	Function	R/W
6–4	ClkSel	Clock Divisor Select	R/W
3	AltVidEn	Alternate VID Change Enable	R/W
2	FidVidEn	FID/VID Change Enable	R/W
1	NBLowPwrEn	Northbridge Low Power Enable	R/W
0	CPULowPwrEn	CPU Low Power Enable	R/W

#### PMM Value Field Descriptions

**Clock Divisor Select (ClkSel)**—Bits 6–4. This field specifies the divisor to use when ramping down the CPU clock or the Northbridge clock.

000b = Divide by 8	100b = Divide by 128
001b = Divide by 16	101b = Divide by 256
010b = Divide by 32	110b = Divide by 512
011b = Divide by 64	111b = reserved

**Alternate VID Change Enable (AltVidEn)**—Bit 3. Enables a VID change to the value programmed in the AltVid field (Function 3, Offset D8h) after CPU clocks and Northbridge clocks are ramped down. This bit should never be set for FID/VID changes or when UMA graphics are used. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

0 = Alternate VID change disabled
1 = Alternate VID change enabled

**FID/VID Change Enable (FidVidEn)**—Bit 2. Enables a change in Frequency ID (FID) and/or Voltage ID (VID). The CPU and the Northbridge clocks must also be ramped down (see NBLowPwrEn and CPULowPwrEn).

0 = FID/VID change disabled
1 = FID/VID change enabled

**Northbridge Low Power Enable (NBLowPwrEn)**—Bit 1. Causes the Northbridge clock to ramp down according to the clock divisor specified in ClkSel and puts the DRAM in self-refresh mode. The CPU clock must also be ramped down (see CPULowPwrEn).

0 = Northbridge low power disabled
1 = Northbridge low power enabled

**CPU Low Power Enable (CPULowPwrEn)**—Bit 0. Causes the CPU clock to ramp down according to the clock divisor specified in ClkSel.



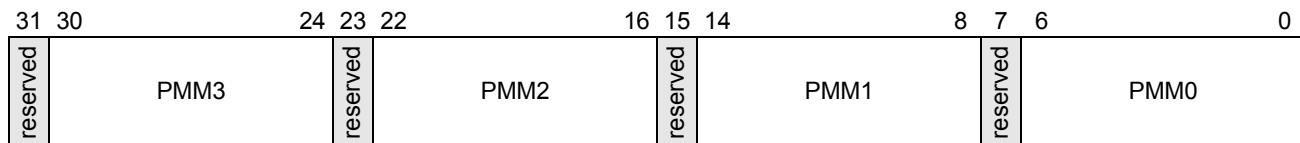
0 = CPU low power disabled

1 = CPU low power enabled

### 3.6.11.2 Power Management Control Low Register

#### Power Management Control Low Register

Function 3: Offset 80h



Bits	Mnemonic	Function	R/W	Reset
31	reserved		R	0
30–24	PMM3	Power Management Mode 3	R/W	0
23	reserved		R	0
22–16	PMM2	Power Management Mode 2	R/W	0
15	reserved		R	0
14–8	PMM1	Power Management Mode 1	R/W	0
7	reserved		R	0
6–0	PMM0	Power Management Mode 0	R/W	0

#### Field Descriptions

**Power Management Mode 0 (PMM0)**—Bits 6–0. See “Power Management Mode (PMM) Value” on page 152 for a description of these bits.

**Power Management Mode 1 (PMM1)**—Bits 14–8. See “Power Management Mode (PMM) Value” on page 152 for a description of these bits.

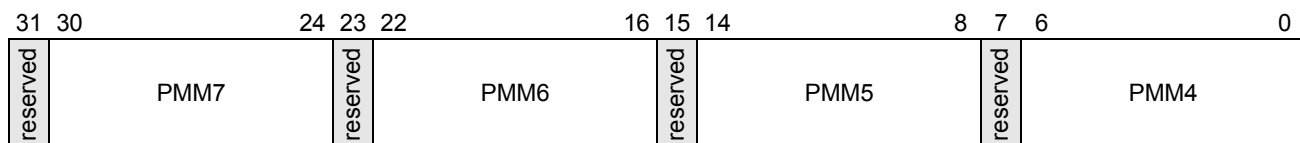
**Power Management Mode 2 (PMM2)**—Bits 22–16. See “Power Management Mode (PMM) Value” on page 152 for a description of these bits.

**Power Management Mode 3 (PMM3)**—Bits 30–24. See “Power Management Mode (PMM) Value” on page 152 for a description of these bits.

### 3.6.11.3 Power Management Control High Register

#### Power Management Control High Register

Function 3: Offset 84h



Bits	Mnemonic	Function	R/W	Reset
31	reserved		R	0
30–24	PMM7	Power Management Mode 7	R/W	0
23	reserved		R	0
22–16	PMM6	Power Management Mode 6	R/W	0
15	reserved		R	0
14–8	PMM5	Power Management Mode 5	R/W	0
7	reserved		R	0
6–0	PMM4	Power Management Mode 4	R/W	0

## Field Descriptions

**Power Management Mode 4 (PMM4)**—Bits 6–0. See “Power Management Mode (PMM) Value” on page 152 for a description of these bits.

**Power Management Mode 5 (PMM5)**—Bits 14–8. See “Power Management Mode (PMM) Value” on page 152 for a description of these bits.

**Power Management Mode 6 (PMM6)**—Bits 22–16. See “Power Management Mode (PMM) Value” on page 152 for a description of these bits.

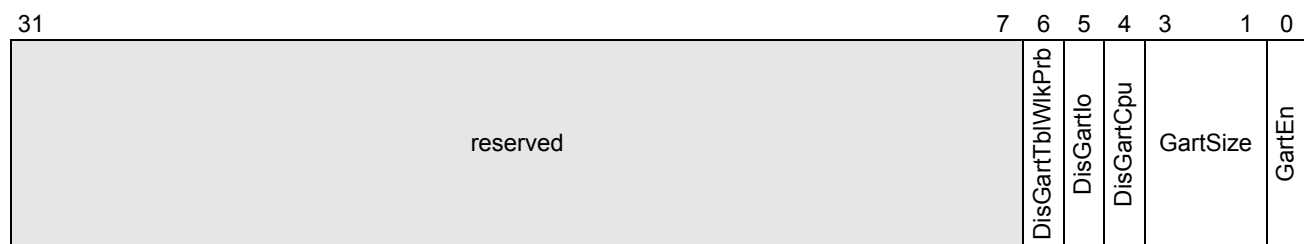
**Power Management Mode 7 (PMM7)**—Bits 30–24. See “Power Management Mode (PMM) Value” on page 152 for a description of these bits.

## 3.6.12 GART Aperture Control Register

This register contains the aperture size and enable for the graphics aperture relocation table (GART) mechanism.

### GART Aperture Control Register

### Function 3: Offset 90h



Bits	Mnemonic	Function	R/W	Reset
31–7	reserved		R	0
6	DisGartTblWlkPrb	Disable GART Table Walk Probes	R/W	0
5	DisGartIo	Disable GART I/O Accesses	R/W	0
4	DisGartCpu	Disable GART CPU Accesses	R/W	0

Bits	Mnemonic	Function	R/W	Reset
3–1	GartSize	GART Size	R/W	0
0	GartEn	GART Enable	R/W	0

## Field Descriptions

**GART Enable (GartEn)**—Bit 0. Enables GART address translation for accesses falling within the GART aperture. GartAperBaseAddr (Function 3, Offset 94h) and other related registers should be initialized before GartEn is set.

**GART Size (GartSize)**—Bits 3–1. Defines the virtual address space to be allocated to the GART.

000b = 32 Mbytes  
 001b = 64 Mbytes  
 010b = 128 Mbytes  
 011b = 256 Mbytes  
 100b = 512 Mbytes  
 101b = 1 Gbyte  
 110b = 2 Gbyte  
 111b = reserved

**Disable GART CPU Accesses (DisGartCpu)**—Bit 4. Disables requests from CPUs from accessing the GART.

**Disable GART I/O Accesses (DisGartIo)**—Bit 5. Disables requests from I/O devices from accessing the GART.

**Disable GART Table Walk Probes (DisGartTblWlkPrb)**—Bit 6. Disables generation of probes for GART table walks. This bit may be set to improve performance in cases where the GART table entries are in address space which is marked uncacheable in processor MTRRs or page tables. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

### 3.6.13 GART Aperture Base Register

This register contains the base address of the aperture for the graphics aperture relocation table (GART). The GART aperture base along with the GART aperture size define the GART aperture address window. BIOS can place the GART aperture below the 4-gigabyte level in address space in order to support legacy operating systems and legacy AGP cards (that do not support 64-bit address space).

#### GART Aperture Base Register

Function 3: Offset 94h

31	15	14	0
reserved			GartAperBaseAddr[39:25]

Bits	Mnemonic	Function	R/W	Reset
31–15	reserved		R	0
14–0	GartAperBaseAddr[39:25]	GART Aperture Base Address Bits 39–25	R/W	X

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

## Field Descriptions

**GART Aperture Base Address Bits 39–25 (GartAperBaseAddr[39:25])**—Bits 14–0. These bits are used to compare to the incoming addresses to determine if they are in the aperture range. They are active based on the aperture size. The remaining address bits are assumed to be 0.

[39:25] = 32 Mbytes

[39:26] = 64 Mbytes

[39:27] = 128 Mbytes

[39:28] = 256 Mbytes

[39:29] = 512 Mbytes

[39:30] = 1 Gbytes

[39:31] = 2 Gbytes

### 3.6.14 GART Table Base Register

This register contains the base address of the translation table for the graphics aperture relocation table (GART). The GART table base points to the base address of a table of 32-bit GART page table entries (PTEs) that contain the physical addresses to use for an incoming address that falls within the GART aperture.

#### GART Table Base Register

#### Function 3: Offset 98h

31	4	3	0
GartTblBaseAddr[39:12]			reserved

Bits	Mnemonic	Function	R/W	Reset
31–4	GartTblBaseAddr[39:12]	GART Table Base Address Bits 39–12	R/W	X
3–0	reserved		R	0

"X" in the Reset column indicates that the field initializes to an undefined state after reset.

## Field Descriptions

**GART Table Base Address Bits 39–12 (GartTblBaseAddr[39:12])**—Bits 31–4. These bits point to the base of the table of GART PTEs to be used for GART address translation. Table 38 shows the organization of each 32-bit PTE in the GART table.

**Table 38. GART PTE Organization**

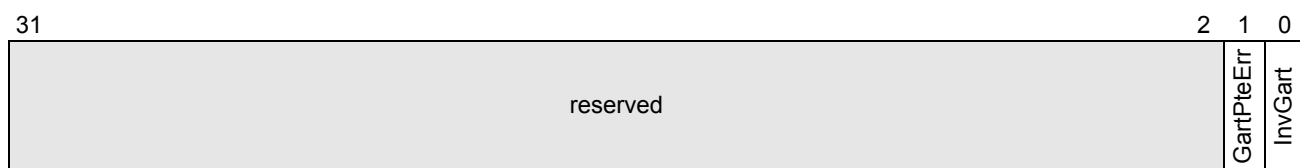
Bits	Description
0	Valid
1	Coherent
3–2	reserved
11–4	PhysAddr[39:32]
31–12	PhysAddr[31:12]

### 3.6.15 GART Cache Control Register

This register controls the GART cache, which is used to cache the most recently translated GART aperture accesses. The GART cache is enabled automatically whenever the GART aperture is enabled.

#### GART Cache Control Register

Function 3: Offset 9Ch



Bits	Mnemonic	Function	R/W	Reset
31–2	reserved		R	0
1	GartPteErr	GART PTE Error	R/WC	0
0	InvGart	Invalidate GART	R/W	0

#### Field Descriptions

**Invalidate GART (InvGart)**—Bit 0. Setting this bit causes the GART cache to be invalidated. This bit is cleared by hardware when the invalidation is complete.

**GART PTE Error (GartPteErr)**—Bit 1. This bit is set when an invalid PTE is encountered during a table walk. Cleared by writing a 1.

### 3.6.16 Clock Power/Timing Low Register

This register controls the transition times between various voltage and frequency states. The value of this register is maintained through a warm reset and is initialized to 0 on a cold reset.

**Clock Power/Timing Low Register****Function 3: Offset D4h**

31	16	15	12	11	8	7	6	4	3	2	0
LCIkPLLLock (19–4)					ReConDel	ClkRampHyst	reserved			GPE	

Bits	Mnemonic	Function	R/W	Reset
31–16	LCIkPLLLock	HyperTransport CLK PLL Lock Counter Bits 19–4	R/W	0
15–12	ReConDel	Link Reconnect Delay	R/W	Ah
11–8	ClkRampHyst	Clock Ramp Hysteresis	R/W	0
7–3	reserved		R	0
2–0	GPE	Good Phase Error	R/W	0

**Field Descriptions**

**Good Phase Error (GPE)**—Bits 2–0. This field defines the BCLK PLL time until good phase error. Counting occurs when at full frequency.

Revision B and earlier revision encodings are:

000b = 16 system clocks  
 001b = 400 system clocks  
 010b = 800 system clocks  
 011b = 1200 system clocks  
 100b = 1600 system clocks  
 101b = 2000 system clocks  
 110b = 2400 system clocks  
 111b = 4095 system clocks

Revision C encodings are:

000b = reserved  
 001b = 200 system clocks (1us)  
 010b = 400 system clocks (2us)  
 011b = 600 system clocks (3us)  
 100b = 800 system clocks (4us)  
 101b = 1600 system clocks (8us)  
 110b = 3000 system clocks (15us)  
 111b = 20000 system clocks (100us)

**Clock Ramp Hysteresis (ClkRampHyst)**—Bits 11–8. A non-zero value in this field enables a hysteresis time which prevents the CPU clock grid from being ramped down after processing a probe. It avoids unnecessary changes of the CPU clock grid when the probe arrival rate is

relatively low. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

Revision D and earlier revisions encodings are:

000b = 0  
 001b = 125 ns  
 010b = 250 ns  
 011b = 375 ns  
 100b = 500 ns  
 101b = 750 ns  
 110b = 1000 ns  
 111b = 2000 ns

Revision E and later revisions encodings are:

0000b = 0  
 0001b = 125 ns  
 0010b = 250 ns  
 0011b = 375 ns  
 0100b = 500 ns  
 0101b = 750 ns  
 0110b = 1000 ns  
 0111b = 2000 ns  
 1000b = 4  $\mu$ s  
 1001b = 8  $\mu$ s  
 1010b = 16  $\mu$ s  
 1011b = 32  $\mu$ s  
 1100b = Reserved  
 1101b = Reserved  
 1110b = Reserved  
 1111b = Reserved

**Link Reconnect Delay (ReConDel)**—Bits 12-15. A non-zero value in this field specifies the approximate delay, in microseconds, from the deassertion of LDTSTOP\_L until the link initialization process is allowed to start. This delay is only applied if LdtStopTriEn = 1 and LdtStopTriClkOvr = 0 (Function 0, Offsets 84h, Ah4, C4h). See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

0000b = 0  
 0001b = 1  $\mu$ s  
 0010b = 2  $\mu$ s  
 0011b = 3  $\mu$ s  
 0100b = 4  $\mu$ s  
 0101b = 5  $\mu$ s  
 0110b = 6  $\mu$ s  
 0111b = 7  $\mu$ s

1000b= 8  $\mu$ s  
 1001b= 9  $\mu$ s  
 1010b= 10  $\mu$ s  
 1011b= Reserved  
 1100b= Reserved  
 1101b= Reserved  
 1110b= Reserved  
 1111b= Reserved

**HyperTransport CLK PLL Lock Counter Bits 19–4 (LClkPLLLock)**—Bits 31–16. This bit field indicates how long it takes for the slowest HyperTransport technology clock PLL to ramp to its new frequency and lock.

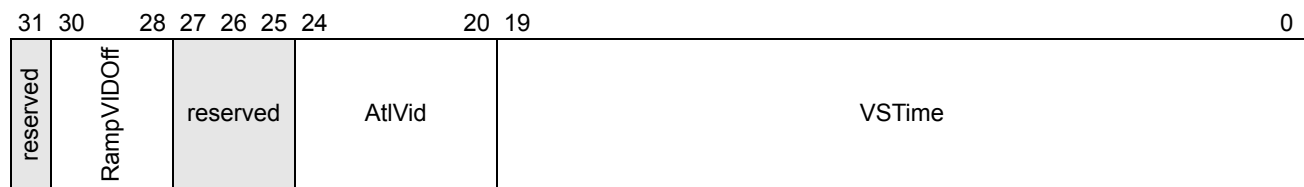
The 16 bits represent the most significant 16 bits of a 20-bit value. The number reflects the number of system bus clocks (5 ns) to wait. It is only necessary to wait a short amount of time in the event the frequency change is one that maintains the VCO frequency. In this case, the PLL will be re-locked quickly.

### 3.6.17 Clock Power/Timing High Register

This register controls the transition times between various voltage and frequency states. The value of this register is maintained through a warm reset and is initialized to 0 on a cold reset.

#### Clock Power/Timing High Register

#### Function 3: Offset D8h



Bits	Mnemonic	Function	R/W	Reset
31	reserved		R	0
30–28	RampVIDOff	Ramp VID Offset	R/W	0
27	reserved		R/W	0
26–25	reserved		R	0
24–20	AltVid	Alternate VID	R/W	0
19–0	VSTime	Voltage Regulator Stabilization Time	R/W	0

#### Field Descriptions

**Voltage Regulator Stabilization Time (VSTime)**—Bits 19–0. This field indicates when the voltage regulator is stable at the Ramp VID before ramping up the clocks. The count is the number of 5-ns system clock cycles.



**Alternate VID (AltVid)**—Bits 24–20. This field specifies the alternate VID while in low power states when enabled through PMM fields in the Power Management Control registers. Switching to AltVID can only be initiated by a StpClk message from the I/O Hub.

If an attempt is made to write a VID value that corresponds to a voltage greater than the voltage that MaxVID corresponds to in the FIDVID\_STATUS register then the MaxVID value is written instead. If an attempt is made to write a VID value that corresponds to a voltage lower than the voltage that MinVID corresponds to in the FIDVID\_STATUS register then the MinVID value is written instead. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this register.

**Ramp VID Offset (RampVIDOff)**—Bits 30–28. Defines the amount of extra voltage required while the PLL is ramping for low power states. This “over-voltage” is applied only until the PLL has phase locked to within specifications.

000b = 0 mV  
 001b = 25 mV  
 010b = 50 mV  
 011b = 75 mV  
 100b = 100 mV  
 101b = 125 mV  
 110b = 150 mV  
 111b = 175 mV

### 3.6.18 HyperTransport™ FIFO Read Pointer Optimization Register

This register allows the separation of read/write pointers in the HyperTransport technology receive/transmit FIFOs to be changed from their default settings. The pointer separation written to this register takes effect after a warm reset. The value of this register is maintained through a warm reset and is initialized to 0 on a cold reset.

#### HyperTransport™ FIFO Read Pointer Optimization Register      Function 3: Offset DCh

31	22	21	20	19	18	16	15	14	13	12	11	10	8	7	6	5	4	3	2	0
reserved				XmtRdPtrLdt2	reserved	RcvRdPtrLdt2	reserved	XmtRdPtrLdt1	reserved	RcvRdPtrLdt1	reserved	XmtRdPtrLdt0	reserved	RcvRdPtrLdt0						

Bits	Mnemonic	Function	R/W	Reset
31–22	reserved		R	0
21–20	XmtRdPtrLdt2	Change Read Pointer For HyperTransport Link 2 Transmitter	R/W	0
19	reserved		R	0

Bits	Mnemonic	Function	R/W	Reset
18–16	RcvRdPtrLdt2	Change Read Pointer For HyperTransport Link 2 Receiver	R/W	0
15–14	reserved		R	0
13–12	XmtRdPtrLdt1	Change Read Pointer For HyperTransport Link 1 Transmitter	R/W	0
11	reserved		R	0
10–8	RcvRdPtrLdt1	Change Read Pointer For HyperTransport Link 1 Receiver	R/W	0
7–6	reserved		R	0
5–4	XmtRdPtrLdt0	Change Read Pointer For HyperTransport Link 0 Transmitter	R/W	0
3	reserved		R	0
2–0	RcvRdPtrLdt0	Change Read Pointer For HyperTransport Link 0 Receiver	R/W	0

## Field Descriptions

**Change Read Pointer For HyperTransport Link 0 Receiver (RcvRdPtrLdt0)**—Bits 2–0. See RcvRdPtrLdt2 for values.

**Change Read Pointer For HyperTransport Link 0 Transmitter (XmtRdPtrLdt0)**—Bits 5–4. See XmtRdPtrLdt2 for values.

**Change Read Pointer For HyperTransport Link 1 Receiver (RcvRdPtrLdt1)**—Bits 10–8. See RcvRdPtrLdt2 for values.

**Change Read Pointer For HyperTransport Link 1 Transmitter (XmtRdPtrLdt1)**—Bits 13–12. See XmtRdPtrLdt2 for values.

**Change Read Pointer For HyperTransport Link 2 Receiver (RcvRdPtrLdt2)**—Bits 18–16. Moves the read pointer for the HyperTransport receive FIFO closer to the write pointer thereby reducing latency through the receiver.

000b = RdPtr assigned by hardware

001b = Move RdPtr closer to WrPtr by 1 HyperTransport clock period

010b = Move RdPtr closer to WrPtr by 2 HyperTransport clock periods

011b = Move RdPtr closer to WrPtr by 3 HyperTransport clock periods

100b = Move RdPtr closer to WrPtr by 4 HyperTransport clock periods

101b = Move RdPtr closer to WrPtr by 5 HyperTransport clock periods

110b = Move RdPtr closer to WrPtr by 6 HyperTransport clock periods

111b = Move RdPtr closer to WrPtr by 7 HyperTransport clock periods

AMD recommends setting this field to 5 for all coherent HyperTransport links and noncoherent HyperTransport links to AMD chipsets. Optimal value for noncoherent HyperTransport links to other chipsets needs to be determined by the developer and tested to ensure stability.

**Change Read Pointer For HyperTransport Link 2 Transmitter (XmtRdPtrLdt2)—Bits 21–20.**

Moves the read pointer for the HyperTransport technology transmit FIFO closer to the write pointer thereby reducing latency through the transmitter.

00b = RdPtr assigned by hardware

01b = Move RdPtr closer to WrPtr by 1 HyperTransport clock period

10b = Move RdPtr closer to WrPtr by 2 HyperTransport clock periods

11b = Move RdPtr closer to WrPtr by 3 HyperTransport clock periods

AMD recommends setting this field to 2 for all coherent HyperTransport links and noncoherent HyperTransport links to AMD chipsets. Optimal value for noncoherent HyperTransport links to other chipsets needs to be determined by the developer and tested to ensure stability.

**3.6.19 Thermtrip Status Register**

The Thermtrip Status register provides status information regarding the THERMTRIP thermal sensor.

**Thermtrip Status Register****Function 3: Offset E4h**

31	30	29	28	25	24	23	14	13	8	7	6	5	4	3	2	1	0		
SwThermtp	reserved		TCaseMax	DiodeOffsetSignBit	reserved				DiodeOffset				reserved	ThermtpEn	ThermtpSense1	ThermtpSense0	reserved	Thermtp	reserved

Bits	Mnemonic	Function	R/W	Reset
31	SwThermtp	Software Thermtrip	W	0
30–29	reserved		R	0
28–25	TCaseMax	Case Temperature Specification	R	0
24	DiodeOffsetSignBit	Diode Offset Sign Bit	R	
23–14	reserved		R	0
13–8	DiodeOffset	Diode Offset	R	
7–6	reserved		R	0
5	ThermtpEn	Thermtrip Enabled	R	
4	ThermtpSense1	Thermtrip Sense 1	R	
3	ThermtpSense0	Thermtrip Sense 0	R	
2	reserved		R	0
1	Thermtp	Thermtrip	R	
0	reserved		R	0

**Field Descriptions**

**Thermtrip (Thermtp)—Bit 1.** Set to 1 if a temperature sensor trip occurs and was enabled.

**Thermtrip Sense 0 (ThermtpSense0)**—Bit 3. Set to 1 if a temperature sensor trip occurs on core 0. The value of this bit is maintained through warm reset.

**Thermtrip Sense 1 (ThermtpSense1)**—Bit 4. Set to 1 if a temperature sensor trip occurs on core 1. The value of this bit is maintained through warm reset.

**Thermtrip Enabled (ThermtpEn)**—Bit 5. Indicates that the thermtrip temperature sensor is enabled. When this bit is set to 1, a THERMTRIP High event will cause the hardware to shut down the PLL, assert the THERMTRIP output pin and set the ThermtpHi bit. The ThermtpSense bit is set for a THERMTRIP High event, irrespective of the state of ThermtpEn.

**Diode Offset (DiodeOffset[5:0])**—Bits 13–8. Thermal diode offset is used to correct the measurement made by an external temperature sensor. This diode offset supports temperature sensors using two sourcing currents only. Other sourcing current implementations are not compatible with the diode offset and are not supported by AMD. The allowable offset range is provided in the appropriate processor functional data sheet, and the maximum offset can vary for different processors. A correction to the offset may be needed for some temperature sensors. Contact the temperature sensor vendor to determine whether an offset correction is needed.

With the diode offset, the thermal diode can be used to ensure the processor is within its functional temperature limits. When the thermal diode measurement minus diode offset equals the maximum control temperature ( $T_{\text{CONTROL max}}$ ), the processor has reached its case temperature specification ( $T_{\text{CASE max}}$ ). The maximum control temperature is provided in the appropriate power and thermal data sheet. The relationship between  $T_{\text{CASE max}}$  and  $T_{\text{CONTROL max}}$  is described in the appropriate functional data sheet. The case temperature specification is provided in the THERMTRIP Status Register and is discussed below.

This field defines a 6 bit unsigned value between the range of 0 to 63 C. The diode offset should be subtracted from the temperature sensor reading. For revision D and later revisions the DiodeOffsetSignBit determines if the diode offset should be added to (DiodeOffsetSignBit=1, negative offset temperature) or subtracted from (DiodeOffsetSignBit=0, positive offset temperature) the temperature sensor reading.

See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Table 39. Diode Offset Encodings**

Offset Temperature	Diode Offset Revision CG and Earlier	Diode Offset Revision D and Later	Diode Offset Sign Bit
-63°C	N/A	111111b	1b
-62°C	N/A	111110b	1b
...	...	...	...
-1°C	N/A	000001b	1b

**Table 39. Diode Offset Encodings**

Offset Temperature	Diode Offset Revision CG and Earlier	Diode Offset Revision D and Later	Diode Offset Sign Bit
0°C	000000b	000000b	0b
1°C	000001b	000001b	0b
...	...	...	...
62°C	111110b	111110b	0b
63°C	111111b	111111b	0b

**Diode Offset Sign Bit (DiodeOffsetSignBit)**—Bit 24. This bit determines if the DiodeOffset field is a negative or positive temperature. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

0 = positive temperature

1 = negative temperature

**Case Temperature Specification (TCaseMax)**—Bits 28-25. This field provides the case temperature specification for the processor. The case temperature specification is calculated multiplying this field by 2 and adding 49. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

0000b = Not used. See the power and thermal datasheet for T<sub>CASE</sub> max specification.

0001b = 51°C

...

1111b = 79°C

**Software Thermtrip (SwThermtp)**—Bit 31. Writing a 1 to this bit position induces a THERMTRIP event. This bit is write-only and returns 0 when read. This is a diagnostic bit, and it should be used for testing purposes only.

### 3.6.20 Northbridge Capabilities Register

The Northbridge Capabilities register indicates whether or not this Northbridge is capable of certain behavior.

#### Northbridge Capabilities Register

#### Function 3: Offset E8h

31	14	13	12	11	9	8	7	6	5	4	3	2	1	0
reserved				CmpCap	reserved		MemCntCap	reserved	DramFreq	ChipKillEccCap	EccCap	BigMPCap	MPCap	128BitCap

Bits	Mnemonic	Function	R/W	Reset
31–14	reserved		R	0
13–12	CmpCap	Dual-Core Capability	R	
11–9	reserved		R	0
8	MemCntCap	Memory Controller Capable	R	
7	reserved		R	0
6–5	DramFreq	Maximum DRAM Frequency	R	
4	ChipKillEccCap	Chip-Kill ECC Capable	R	
3	EccCap	ECC Capable	R	
2	BigMPCap	Big MP Capable	R	
1	MPCap	MP Capable	R	
0	128BitCap	128-Bit DRAM Capable	R	

## Field Descriptions

**128-Bit DRAM Capable (128BitCap)**—Bit 0. This bit is set to 1 if the Northbridge is capable of supporting a 128-bit DRAM interface.

**MP Capable (MPCap)**—Bit 1. This bit is set to 1 if the Northbridge is capable of supporting multiprocessor systems.

**Big MP Capable (BigMPCap)**—Bit 2. This bit is set to 1 if the Northbridge is capable of supporting multiprocessor systems greater than DP.

**ECC Capable (EccCap)**—Bit 3. This bit is set to 1 if the Northbridge is capable of supporting ECC.

**Chip-Kill ECC Capable (ChipKillEccCap)**—Bit 4. This bit is set to 1 if the Northbridge is capable of supporting chip-kill ECC.

**Maximum DRAM Frequency (DramFreq)**—Bits 6–5. Indicates the maximum DRAM frequency supported.

- 00b = No limit
- 01b = 166 MHz
- 10b = 133 MHz
- 11b = 100 MHz

**Memory Controller Capable (MemCntCap)**—Bit 8. This bit is set to 1 if the Northbridge is capable of supporting an on-chip memory controller.

**Dual-Core Capability (CmpCap)**—Bits 13–12. Indicates the number of CPU cores present in the processor. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this register.

- 00b = Single core processor
- 01b = Dual core processor
- 10b = Reserved
- 11b = Reserved







## 4 DRAM Configuration

---

### 4.1 Programming Interface

This section describes how to program various DRAM controller registers. There are two principal areas of configuration:

- Configuration state information obtained via the DIMM Serial Presence Detect (SPD) ROMs. Non-SPD memory sizing is not supported.
- All other configuration state information.

#### 4.1.1 SPD ROM-Based Configuration

The SPD device is an EEPROM on the DIMM encoded by the DIMM manufacturer. The description of the EEPROM is usually provided on a data sheet for the DIMM itself along with data describing the memory devices used. The data describes configuration and speed characteristics of the DIMM and the SDRAM components mounted on the DIMM. The data sheet also contains the DIMM byte values that are encoded in the SPD on the DIMM.

BIOS acquires the values encoded in the SPD ROM through the I/O hub, which obtains the information through a secondary device connected to the I/O hub through the SMBus. This secondary device communicates with the DIMM by means of the I<sup>2</sup>C bus.

The SPD ROM provides values for several DRAM timing parameters that are required by the DRAM controller. These parameters are:

- tCL: (CAS latency)
- tRC: Active-to-Active/Auto Refresh command period
- tRFC: Auto-Refresh-to-Active/Auto Refresh command period
- tRCD: Active-to-Read-or-Write delay
- tRRD: Active-Bank-A to-Active-Bank-B delay
- tRAS: Active-to-Precharge delay
- tRP: Precharge time
- tREF: Refresh interval (function of DRAM density)

##### 4.1.1.1 tCL (CAS Latency)

The number of memory clocks it takes a DRAM to return data after the read CAS\_L is asserted depends on the memory clock frequency. The value that BIOS programs into the memory controller is

a function of the target clock frequency. The target clock frequency is determined from the supported CAS latencies at given clock frequencies of each DIMM. A suggested algorithm is as follows:

1. Determine all CAS latencies supported by each installed DIMM, defined in SPD byte 18. One bit corresponds to each supported CAS latency. SPD byte 18 specifies CAS latencies with which devices on the DIMM can reliably operate. BIOS should not assume that a device can operate with either slower or faster CAS latency than those specified by the SPD. Typically, all DDR DIMMs support CAS latencies of 2 and 2.5. Some DDR DIMMs may support CAS latencies of 3. The DRAM controller is designed to support these CAS latencies. The SPD ROM allows the manufacturer to indicate support for CAS latency 3.5. This value is not supported by the DRAM controller and should be discarded.
2. Determine the maximum clock frequency at each supported CAS latency for each DIMM. The minimum cycle time for the highest, the second highest, and the third highest supported CAS latencies is defined in SPD byte 9, 23, and 25, respectively. There is a possibility of incorrectly programmed SPDs such that a cycle time from SPD byte 23 or 25, corresponding to a set bit in SPD byte 18, is unimplemented. BIOS should discard this pair. The minimum cycle time has 1/10ns granularity.
3. Determine the best CAS latency and clock frequency combination. Find the highest clock frequency supported by the slowest DIMM and determine the CAS latency at that operating frequency. It is necessary to choose the highest CAS latency supported by all the DIMMs at the target frequency.

There is a possibility that there are two options: a 166 MHz tCL=3 configuration or a 133 MHz tCL=2 configuration. The above algorithm would result in selecting the 166 MHz configuration, but it is known through performance analysis that this is not the best choice. Whenever tCL for the higher frequency is 1 greater than tCL for the next lowest frequency, BIOS should select the lower frequency. If the tCL difference is 0.5, then the higher frequency should be selected.

#### 4.1.1.2 tRCD (RAS-to-CAS Delay)

This parameter is defined in SPD byte 29 and it has 1/4ns granularity. BIOS should read and convert this value into a number of DRAM clocks using the target frequency obtained in “tCL (CAS Latency)” on page 169. Typically, this value is 48h for DDR333, which maps to 18 ns or 3 clocks.

#### 4.1.1.3 tRAS (Active-to-Precharge Delay)

This parameter is defined in SPD byte 30 and it has 1-ns granularity. BIOS should read and convert this value into a number of DRAM clocks using the target frequency obtained in “tCL (CAS Latency)” on page 169. Typically, this value is 2Ah for DDR333, which maps to 42 ns or 7 clocks.

#### 4.1.1.4 tRP (Precharge Command Period)

This parameter is defined in SPD byte 27 and it has 1/4-ns granularity. BIOS should read and convert this value into a number of DRAM clocks using the target frequency obtained in “tCL (CAS Latency)” on page 169. Typically, this value is 48h for DDR333, which maps to 18 ns or 3 clocks.

#### 4.1.1.5 tRC (Active-to-Active/Auto-Refresh Command Period)

This parameter is defined in SPD byte 41 and it has 1-ns granularity. BIOS should read and convert this value into a number of DRAM clocks using the target frequency obtained in “tCL (CAS Latency)” on page 169. Typically, this value is 3Ch for DDR333, which maps to 60 ns or 10 clocks.

Some DIMMs may not include the tRC value in byte 41. In this case, BIOS will read either 00h or FFh from the ROM and it should do the following:

- At 100 MHz, tRC is assumed to be 46h, which maps to 70 ns or 7 clocks.
- At 133 MHz, tRC is assumed to be 41h, which maps to 65 ns or 9 clocks.
- At 166 MHz, tRC is assumed to be 3Ch, which maps to 60 ns or 10 clocks.
- At 200 MHz, tRC is assumed to be 37h, which maps to 55 ns or 11 clocks.

#### 4.1.1.6 tRRD (Active-to-Active of a Different Bank)

This parameter is defined in SPD byte 28 and it has 1/4-ns granularity. BIOS should read this value and convert into a number of DRAM clocks using the target frequency obtained in “tCL (CAS Latency)” on page 169. Typically, this value is 30h for DDR333, which maps to 12 ns or 2 clocks.

#### 4.1.1.7 tRFC (Auto-Refresh-to-Active/Auto-Refresh Command Period)

This parameter is defined in SPD byte 42 and it has 1-ns granularity. BIOS should read and convert this value into a number of DRAM clocks using the target frequency obtained in “tCL (CAS Latency)” on page 169. Typically, this value is 48h for DDR333 DRAMs between 64-Mbit and 512-Mbit and 78h for DDR333 1-Gbit DRAMS.

Some DIMMs may not include the tRFC value in byte 42. In this case, BIOS will read either 00h or FFh from the ROM and it should do the following:

- At 100 MHz, tRFC is assumed to be 50h, which maps to 80 ns or 8 clocks.
- At 133 MHz, tRFC is assumed to be 4Bh, which maps to 75 ns or 10 clocks.
- At 166 MHz, tRFC is assumed to be 48h, which maps to 72 ns or 12 clocks for 512-Mbit devices or smaller. For 1-Gbit devices, it is assumed to be 78h, which maps to 120 ns or 20 clocks.
- At 200 MHz, tRFC is assumed to be 46h, which maps to 70 ns or 14 clocks for 512-Mbit or smaller devices. For 1-Gbit devices, it is assumed to be 78h, which maps to 120 ns or 24 clocks.

#### 4.1.1.8 tREF (Refresh Rate)

The tREF is not an SPD ROM parameter but a field that the DRAM controller requires and that can be obtained from several bytes in the SPD ROM. Typically, the entire DRAM must have every row refreshed at least once every 64 ms. The average time between two row refreshes is thus based on the number of rows in the DRAM and the clock frequency.

The clock frequency was previously derived. The number of rows in the implementation can be read from SPD byte 3. This provides the number of row address bits for each of the two possible chip selects on a DIMM. (Note that each chip-select range of a two chip-select DIMM uses devices that have the same number of rows). A design with 12 row address bits implies 4k rows per internal device chip select. This maps to a refresh every 15.6  $\mu$ s on average. There are also 8k (13 row address bits) and 16k (14 row address bits) row devices. By knowing the number of rows and the frequency, the DRAM controller configuration register can be correctly programmed. All 4k row devices require an average refresh interval of 15.6  $\mu$ s and all 8k and 16k row devices require a 7.8  $\mu$ s average refresh interval.

#### 4.1.1.9 Registered or Unbuffered DIMMs

SPD byte 21, bit 1 indicates whether the DIMM registers the address and commands or not. Refer to the processor data sheet to determine the type of DIMMs supported. Systems with mixed unbuffered and registered DIMMs are not supported.

#### 4.1.1.10 DIMM Chip Select Density

The size of a DIMM module chip-select range can be obtained directly from the SPD ROM.

SPD byte 5 indicates whether the DIMM is composed of one or two chip selects. Byte 31 indicates the size of the chip-select range or ranges on the DIMM (from 32 Mbyte to 2 Gbyte). See the SPD ROM specification for the encoding.

***Note:** If the DIMM uses two chip selects, then each chip-select range is the same size.*

#### 4.1.1.11 DIMM ECC Enable

SPD byte 11 indicates whether the DIMM supports ECC bits. A value of 02h indicates that ECC is supported for all chip-select banks on the DIMM.

#### 4.1.1.12 x4 DIMMs

The DRAM device width is largely inconsequential to the controller except when they are x4 devices. The width of the devices on the DIMM is determined by the value of SPD byte 13. If the DIMM implements two chip-select banks, then each chip-select bank uses devices of the same width. See the SPD ROM specification for the encoding.

### 4.1.2 Non-SPD ROM-Based Configuration

Many other bit fields are also required by the DRAM controller configuration registers, but these values cannot be obtained from the SPD ROM. In many cases, these values will be hardcoded into the BIOS, but in others they are functions of other bit values. This section describes how BIOS programs each non-SPD related field that is not hardcoded.

#### 4.1.2.1 Twr (Write Recovery)

This is a true DRAM device timing parameter but it is not included in the SPD ROM. Therefore, it will be hardcoded based on the DDR200, DDR266, DDR333, and DDR400 AC specification, as follows:

- DDR200, DDR266: Twr is 2 clocks.
- DDR333, DDR400: Twr is 3 clocks.

#### 4.1.2.2 Twtr (Write to Read Delay)

This parameter has the following values:

- DDR200, DDR266, DDR333: Twtr is 1 clock.
- DDR400: Twtr is 2 clocks.

#### 4.1.2.3 Trwt

This field ensures read-to-write data-bus turnaround. This field is a function of CAS latency and clock frequency. It is also a function of the asynchronous round-trip loop delay for each of the systems shown in Table 40. These settings are conservative initial recommendations meant to account for different board layouts. These values can be adjusted to compensate for device loading or any other empirical data that proves that other settings are more reliable.

**Table 40. Trwt Values**

CAS Latency	System	Number of Clocks			
		200 MHz	166 MHz	133 MHz	100 MHz
CL = 2	128-bit interface	N/A	3	2	2
	64-bit interface (registered DIMMS)	N/A	2	2	2
	64-bit interface (unbuffered DIMMS)	3	3	3	3
CL = 2.5	128-bit interface	N/A	3	3	3
	64-bit interface (registered DIMMS)	N/A	3	3	3
	64-bit interface (unbuffered DIMMS)	4	4	4	4
CL = 3	128-bit interface	3	4	3	3
	64-bit interface (registered DIMMS)	3	3	3	3
	64-bit interface (unbuffered DIMMS)	4	4	4	4

#### 4.1.2.4 Twcl (Write CAS Latency)

If the DIMM is registered, the write CAS latency is two clocks. If the DIMM is unbuffered, then write CAS latency is one clock.

#### 4.1.2.5 Maximum Asynchronous Latency

AsyncLat field should be set according to the round trip loop time from the processor to the farthest DIMM in the design. The round trip loop time is primarily influenced by the bus length of the traces to the farthest DIMM, but is also influenced by device loading and bus frequency. The AsyncLat field should be set to 9 ns for registered DIMM systems with a DDR266 8-DIMM configuration, to 8 ns for registered DIMM systems with a DDR333 8-DIMM configuration, and 8ns for 4 DIMM registered DIMM systems. AsyncLat field should be set to 7 ns for unbuffered systems with 3 DIMMs or 4 DIMMs, and to 6 ns for all other unbuffered DIMM systems. These settings are conservative initial recommendations meant to account for different board layouts. These values can be adjusted to compensate for signal integrity issues, device loading, or any other empirical data that proves that other settings are more reliable. For more information on how to calculate a specific AsyncLat see the relevant motherboard design guide.

#### 4.1.2.6 Read Preamble Time

The Read Preamble time is a function of the asynchronous round trip loop latency, the clock frequency, DIMM type and process type. These settings are initial recommendations only. They can be adjusted to compensate for signal integrity issues, bus length mismatches, or any other empirical data that proves that other settings are more reliable.

**Table 41. RdPreamble Values**

	200 MHz	166 MHz	133 MHz	100 MHz
Registered DIMM slots	7 ns	7.5 ns	8 ns	9 ns
Unbuffered 1 or 2 DIMM slots <sup>1</sup>	5 ns	6 ns	7 ns	9 ns
Unbuffered 3 DIMM slots <sup>1</sup>	5.5 ns	6.5 ns	7.5 ns	9 ns
Unbuffered 4DIMM slots <sup>1</sup>	5.5 ns	6.0 ns	7 ns	9 ns
<b>Notes:</b> 1. The values for RdPreamble must be set based on the number of DIMM slots, independent of the number of DIMMs actually populated. This allows the memory controller to compensate for worst-case round-trip delay from DIMMs in the farthest slot.				

#### 4.1.2.7 Memory Clock Enable

DRAM clocks must be enabled by BIOS. There are four clock enable bits which correspond on a one-to-one basis with the DIMMs (i.e., if DIMM0 is populated, memory clock 0 must be enabled).

DRAM clocks are grouped according to the requirements for unbuffered or registered DIMMs. For example, when MC0\_EN is set to enable DIMM 0 in an unbuffered DIMMs configuration, the memory controller will drive three clock pairs to DIMM 0. In a registered DIMM configuration a single clock pair is driven to each enabled DIMM.

In DIMM configurations that have two DIMM sockets connected to the same command/address bus (as in SODIMM configurations), but have different clock pairs routed to each of the DIMMs, BIOS must enable both sets of clocks even if the DIMM 0 slot is unpopulated. For example, in an SODIMM configuration with two DIMM slots, BIOS should set both MC0\_EN and MC1\_EN even if only DIMM 1 is populated. This is required in order to guarantee proper operation.

### 4.1.3 Maximum DRAM Speed as a Function of Loading

The following tables represent AMD's recommendations based on configurations that have been tested at AMD on specific platforms with a wide variety of DIMMs. Due to the wide variation in platform and DIMM implementations, results may vary, and each motherboard vendor should conduct their own separate validation. Each unique motherboard should be tested to validate the maximum DRAM speeds for the supported DIMM configurations. Following AMD motherboard design guidelines and/or a restricted DIMM vendor list in some cases may allow manufacturers to support higher DRAM speeds than listed in the tables below.

The BIOS can enforce these speeds by reducing the DRAM operating frequency as listed in these tables.

#### 4.1.3.1 754-pin Lidded Micro PGA Package

Some processor pin names in the 754-pin lidded micro PGA package have A and B suffixes for the DDR interface. This is a way to distinguish between two otherwise functionally identical pins which exist as multiple redundant pins to accommodate loading. See the *AMD Athlon™ 64 Processor Motherboard Design Guide*, order# 24665 for details on the valid connection schemes for these functionally redundant pins.

**Table 42. Unbuffered DIMM Support For 754-pin Lidded Micro PGA Package**

Number of DIMMs <sup>5</sup>	DIMM 1 <sup>4.1.5</sup>	DIMM 2 <sup>2</sup>	DIMM 3 <sup>2</sup>	Maximum DRAM Speed	
				1T	2T <sup>3</sup>
1	x8 single rank or x16	empty	empty	DDR400	DDR400
1	empty	x8 single rank or x16	empty	DDR400	DDR400
1	empty	empty	x8 single rank or x16	DDR400	DDR400
1	x8 double rank	empty	empty	DDR400	DDR400
1	empty	x8 double rank	empty	DDR400	DDR400
1	empty	empty	x8 double rank	DDR400	DDR400
2	x8 single rank or x16	x8 single rank or x16	empty	DDR400	DDR400
2	x8 single rank or x16	x8 double rank	empty	DDR400	DDR400
2	x8 single rank or x16	empty	x8 single rank or x16	DDR400	DDR400

**Table 42. Unbuffered DIMM Support For 754-pin Lidded Micro PGA Package**

Number of DIMMs <sup>5</sup>	DIMM 1 <sup>4.1.5</sup>	DIMM 2 <sup>2</sup>	DIMM 3 <sup>2</sup>	Maximum DRAM Speed	
				1T	2T <sup>3</sup>
2	x8 single rank or x16	empty	x8 double rank	DDR400	DDR400
2	x8 double rank	x8 single rank or x16	empty	DDR400	DDR400
2	x8 double rank	x8 double rank	empty	DDR333	DDR333
2	x8 double rank	empty	x8 single rank or x16	DDR400	DDR400
2	x8 double rank	empty	x8 double rank	DDR333	DDR333
2	empty	x8 single rank or x16	x8 single rank or x16	DDR333	DDR400
2	empty	x8 single rank or x16	x8 double rank	DDR200	DDR400
2	empty	x8 double rank	x8 single rank or x16	DDR200	DDR400
2	empty	x8 double rank	x8 double rank	DDR200	DDR333
3	x8 single rank or x16	x8 single rank or x16	x8 single rank or x16	DDR333	DDR400
3	x8 single rank or x16	x8 single rank or x16	x8 double rank	DDR200	DDR333
3	x8 single rank or x16	x8 double rank	x8 single rank or x16	DDR200	DDR333
3	x8 single rank or x16	x8 double rank	x8 double rank	DDR200	DDR333 <sup>4</sup>
3	x8 double rank	x8 single rank or x16	x8 single rank or x16	DDR333	DDR333
3	x8 double rank	x8 single rank or x16	x8 double rank	DDR200	DDR333 <sup>4</sup>
3	x8 double rank	x8 double rank	x8 single rank or x16	DDR200	DDR333 <sup>4</sup>
3	x8 double rank	x8 double rank	x8 double rank	DDR200	DDR333 <sup>4</sup>

- DIMM 1 connects to command/address pins MEMADDA[13:0], MEMBANKA[1:0], MEMRASA\_L, MEMCASA\_L, MEMWEA\_L, MEMCKEA.
- DIMM 2 and 3 connect to command/address pins MEMADDB[13:0], MEMBANKB[1:0], MEMRASB\_L, MEMCASB\_L, MEMWEB\_L, MEMCKEB.
- 2T timing is supported in CG and later silicon revisions. Refer to the AMD Athlon™ 64 Processor Power and Thermal Data Sheet, order #30430, for silicon revision determination.
- The maximum allowable DRAM speed under these high load conditions may be reduced with certain DIMMs due to signal integrity degradation.
- For systems using a 2 DIMM implementation, refer only to rows in this table where the entry for the column titled DIMM 3 reads 'empty'.

#### 4.1.3.2 754-pin Lidless Micro PGA Package

Some processor pin names in the 754-pin lidless micro PGA package have A and B suffixes for the DDR interface. This is a way to distinguish between two otherwise functionally identical pins which



exist as multiple redundant pins to accommodate loading. See the *AMD Athlon™ 64 Processor Motherboard Design Guide*, order# 24665, *Piccolo Schematics (PDF version of OrCAD schematics) Revision 1 File*, and *Piccolo Dual Address/Command Orcad Schematics* for details on the valid connection schemes for these functionally redundant pins. Each motherboard vendor should validate the memory subsystem to determine maximum DRAM speeds for various SO-DIMM configurations.

**Table 43. Unbuffered SO-DIMM Support For 754-pin Lidless Micro PGA Package  
(Single Address/Control Bus Motherboard Implementation<sup>1</sup>)**

Address/Control Loading <sup>2</sup>		Data Loading <sup>3</sup>	Maximum DRAM Speed	
Min	Max		1T	2T <sup>4</sup>
4	18	< 4	DDR400	DDR400
20	26	< 4	DDR333	DDR400
16	34	4	DDR266	DDR266
<ol style="list-style-type: none"> <li>1. SO-DIMM1 and SO-DIMM2 connect to the A copy of the address/control group signals. See the AMD Athlon™ 64 Processor Motherboard Design Guide, order# 24665, and Piccolo Schematics (PDF version of OrCAD schematics) Revision 1 File for details on this connection scheme and other motherboard design rules and recommendations for optimizing DRAM performance.</li> <li>2. Address/Control loading is calculated by summing the electrical loads per bit on the address/control busses presented by all installed SO-DIMMs. Each DRAM IC on an SO-DIMM presents 1 electrical load per address/control bit.</li> <li>3. Data loading is calculated by summing the electrical loads per bit on the data bus presented by all installed SO-DIMMs. A single rank SO-DIMM uses one chip select and presents one electrical load per data bit. A double rank SO-DIMM uses two chip selects and presents two electrical loads per data bit.</li> <li>4. 2T timing is supported in CG and later silicon revisions. Refer to the AMD Athlon™ 64 Processor Power and Thermal Data Sheet, order# 30430, for silicon revision determination.</li> </ol>				

**Table 44. Unbuffered SO-DIMM Support For 754-pin Lidless Micro PGA Package  
(Dual Address/Control Bus Motherboard Implementation<sup>1</sup>)**

Data Loading <sup>2</sup>	Maximum DRAM Speed
< 4	DDR400
4	DDR266
<p>1. SO-DIMM1 connects to the A copy of the address/control group signals and SO-DIMM2 connects to the B copy of the address/control group signals. See the AMD Athlon™ 64 Processor Motherboard Design Guide, order# 24665, and Piccolo Dual Address/Command Orcad Schematics for details on this connection scheme and other motherboard design rules and recommendations for optimizing DRAM performance.</p> <p>2. Data loading is calculated by summing the electrical loads per bit on the data bus presented by all installed SO-DIMMs. A single rank SO-DIMM uses one chip select and presents one electrical load per data bit. A double rank SO-DIMM uses two chip selects and presents two electrical loads per data bit.</p>	

**4.1.3.3 939-pin Lidded Micro PGA Package****Table 45. Unbuffered DIMM Support For 939-pin Lidded Micro PGA Package**

Data Bus	Chip Selects				Maximum DRAM Speed	
	MEMCS_1L_L*	MEMCS_1H_L*	MEMCS_2L_L*	MEMCS_2H_L*	1T	2T
64-bits	Single rank	N/A	Empty	N/A	DDR400	DDR400
	Double rank	N/A	Empty	N/A	DDR400	DDR400
	Empty <sup>1</sup>	N/A	Single rank <sup>1</sup>	N/A	DDR333	DDR333
	Empty	N/A	Double rank	N/A	DDR400	DDR400
	Single rank	N/A	Single rank	N/A	DDR333	DDR400
	Single rank	N/A	Double rank	N/A	DDR200	DDR400
	Double rank	N/A	Single rank	N/A	DDR200	DDR400
	Double rank	N/A	Double rank	N/A	DDR200	DDR333

**Table 45. Unbuffered DIMM Support For 939-pin Lidded Micro PGA Package**

Data Bus	Chip Selects				Maximum DRAM Speed	
	MEMCS_1L_L*	MEMCS_1H_L*	MEMCS_2L_L*	MEMCS_2H_L*	1T	2T
128-bits	Single rank	Single rank	Empty	Empty	DDR400	DDR400
	Double rank	Double rank	Empty	Empty	DDR400	DDR400
	Empty <sup>1</sup>	Empty <sup>1</sup>	Single rank <sup>1</sup>	Single rank <sup>1</sup>	DDR333	DDR333
	Empty	Empty	Double rank	Double rank	DDR400	DDR400
	Single rank	Single rank	Single rank	Single rank	DDR333	DDR400
	Single rank	Single rank	Double rank	Double rank	DDR200	DDR400
	Double rank	Double rank	Single rank	Single rank	DDR200	DDR400
	Double rank	Double rank	Double rank	Double rank	DDR200	DDR333
1. Refer to the AMD Athlon™ 64 939 Processor Motherboard Design Guide, order# 30474, for details on these loading conditions.						

**Table 46. Unbuffered DIMM Support For Revision E 939-pin Lidded Micro PGA Package**

DIMMs	Chip Selects				Maximum DRAM Speed	
	MEMCS_1L_L*	MEMCS_1H_L*	MEMCS_2L_L*	MEMCS_2H_L*	1T	2T
1	Single rank	Empty	Empty	Empty	DDR400	DDR400
	Double rank	Empty	Empty	Empty	DDR400	DDR400
	Empty <sup>1</sup>	Empty	Single rank <sup>1</sup>	Empty	DDR333	DDR333
	Empty	Empty	Double rank	Empty	DDR400	DDR400
	Empty	Single rank	Empty	Empty	DDR400	DDR400
	Empty	Double rank	Empty	Empty	DDR400	DDR400
	Empty	Empty <sup>1</sup>	Empty	Single rank <sup>1</sup>	DDR333	DDR333
	Empty	Empty	Empty	Double rank	DDR400	DDR400
1. Refer to the AMD Athlon™ 64 939 Processor Motherboard Design Guide, order# 30474, for details on these loading conditions.						

**Table 46. Unbuffered DIMM Support For Revision E 939-pin Lidded Micro PGA Package**

DIMMs	Chip Selects				Maximum DRAM Speed	
	MEMCS_1L_L*	MEMCS_1H_L*	MEMCS_2L_L*	MEMCS_2H_L*	1T	2T
2	Single rank	Empty	Single rank	Empty	DDR333	DDR400
	Single rank	Empty	Double rank	Empty	DDR200	DDR400
	Double rank	Empty	Single rank	Empty	DDR200	DDR400
	Double rank	Empty	Double rank	Empty	DDR200	DDR333
	Empty	Single rank	Empty	Single rank	DDR333	DDR400
	Empty	Single rank	Empty	Double rank	DDR200	DDR400
	Empty	Double rank	Empty	Single rank	DDR200	DDR400
	Empty	Double rank	Empty	Double rank	DDR200	DDR333
	Single rank	Single rank	Empty	Empty	DDR400	DDR400
	Double rank	Double rank	Empty	Empty	DDR400	DDR400
	Single rank	Double rank	Empty	Empty	DDR400	DDR400
	Double rank	Single rank	Empty	Empty	DDR400	DDR400
	Empty <sup>1</sup>	Empty <sup>1</sup>	Single rank <sup>1</sup>	Single rank <sup>1</sup>	DDR333	DDR333
	Empty <sup>1</sup>	Empty	Single rank <sup>1</sup>	Double rank	DDR333	DDR333
	Empty	Empty <sup>1</sup>	Double rank	Single rank <sup>1</sup>	DDR333	DDR333
	Empty	Empty	Double rank	Double rank	DDR400	DDR400
1. Refer to the AMD Athlon™ 64 939 Processor Motherboard Design Guide, order# 30474, for details on these loading conditions.						

**Table 46. Unbuffered DIMM Support For Revision E 939-pin Lidded Micro PGA Package**

DIMMs	Chip Selects				Maximum DRAM Speed	
	MEMCS_1L_L*	MEMCS_1H_L*	MEMCS_2L_L*	MEMCS_2H_L*	1T	2T
3	Single rank	Single rank	Single rank	Empty	DDR333	DDR400
	Single rank	Empty <sup>1</sup>	Single rank	Single rank <sup>1</sup>	DDR333	DDR333
	Single rank	Double rank	Single rank	Empty	DDR333	DDR400
	Single rank	Empty	Single rank	Double rank	DDR333	DDR400
	Single rank	Single rank	Double rank	Empty	DDR200	DDR400
	Single rank	Empty <sup>1</sup>	Double rank	Single rank <sup>1</sup>	DDR200	DDR333
	Single rank	Double rank	Double rank	Empty	DDR200	DDR400
	Single rank	Empty	Double rank	Double rank	DDR200	DDR400
	Double rank	Single rank	Single rank	Empty	DDR200	DDR400
	Double rank	Empty <sup>1</sup>	Single rank	Single rank <sup>1</sup>	DDR200	DDR333
	Double rank	Double rank	Single rank	Empty	DDR200	DDR400
	Double rank	Empty	Single rank	Double rank	DDR200	DDR400
	Double rank	Single rank	Double rank	Empty	DDR200	DDR333
	Double rank	Empty <sup>1</sup>	Double rank	Single rank <sup>1</sup>	DDR200	DDR333
	Double rank	Double rank	Double rank	Empty	DDR200	DDR333
	Double rank	Empty	Double rank	Double rank	DDR200	DDR333
	Single rank	Single rank	Empty	Single rank	DDR333	DDR400
	Empty <sup>1</sup>	Single rank	Single rank <sup>1</sup>	Single rank	DDR333	DDR333
	Double rank	Single rank	Empty	Single rank	DDR333	DDR400
	Empty	Single rank	Double rank	Single rank	DDR333	DDR400
	Single rank	Single rank	Empty	Double rank	DDR200	DDR400
	Empty <sup>1</sup>	Single rank	Single rank <sup>1</sup>	Double rank	DDR200	DDR333
	Double rank	Single rank	Empty	Double rank	DDR200	DDR333
	Empty	Single rank	Double rank	Double rank	DDR200	DDR400
	Single rank	Double rank	Empty	Single rank	DDR200	DDR400
	Empty <sup>1</sup>	Double rank	Single rank <sup>1</sup>	Single rank	DDR200	DDR333
	Double rank	Double rank	Empty	Single rank	DDR200	DDR400
	Empty	Double rank	Double rank	Single rank	DDR200	DDR400
	Single rank	Double rank	Empty	Double rank	DDR200	DDR333
	Empty	Double rank	Single rank	Double rank	DDR200	DDR333
	Double rank	Double rank	Empty	Double rank	DDR200	DDR333
	Empty	Double rank	Double rank	Double rank	DDR200	DDR333

1. Refer to the AMD Athlon™ 64 939 Processor Motherboard Design Guide, order# 30474, for details on these loading conditions.

**Table 46. Unbuffered DIMM Support For Revision E 939-pin Lidded Micro PGA Package**

DIMMs	Chip Selects				Maximum DRAM Speed	
	MEMCS_1L_L*	MEMCS_1H_L*	MEMCS_2L_L*	MEMCS_2H_L*	1T	2T
4	Single rank	Single rank	Single rank	Single rank	DDR333	DDR400
	Single rank	Single rank	Single rank	Double rank	DDR200	DDR400
	Single rank	Double rank	Single rank	Single rank	DDR200	DDR400
	Single rank	Double rank	Single rank	Double rank	DDR200	DDR333
	Single rank	Single rank	Double rank	Single rank	DDR200	DDR400
	Single rank	Single rank	Double rank	Double rank	DDR200	DDR400
	Single rank	Double rank	Double rank	Single rank	DDR200	DDR400
	Single rank	Double rank	Double rank	Double rank	DDR200	DDR333
	Double rank	Single rank	Single rank	Single rank	DDR200	DDR400
	Double rank	Single rank	Single rank	Double rank	DDR200	DDR400
	Double rank	Double rank	Single rank	Single rank	DDR200	DDR400
	Double rank	Double rank	Single rank	Double rank	DDR200	DDR333
	Double rank	Single rank	Double rank	Single rank	DDR200	DDR333
	Double rank	Single rank	Double rank	Double rank	DDR200	DDR333
	Double rank	Double rank	Double rank	Single rank	DDR200	DDR333
	Double rank	Double rank	Double rank	Double rank	DDR200	DDR333

1. Refer to the AMD Athlon™ 64 939 Processor Motherboard Design Guide, order# 30474, for details on these loading conditions.

**Table 47. SODIMM Support For Revision E 939-pin Lidded Micro PGA Package**

Data Bus	Chip Selects		Maximum DRAM Speed	
	MEMCS_1L_L*	MEMCS_1H_L*	1T	2T
64-bits	Any	Empty	DDR400	DDR400
	Empty	Any	DDR400	DDR400

**Table 47. SODIMM Support For Revision E 939-pin Lidded Micro PGA Package**

Data Bus	Chip Selects		Maximum DRAM Speed	
	MEMCS_1L_L*	MEMCS_1H_L*	1T	2T
64 bits Mod64BitMux=1	Single rank x8	Single rank x8	DDR400	DDR400
	Single rank x8	Single rank x16	DDR400	DDR400
	Single rank x8	Double rank x8	DDR400	DDR400
	Single rank x8	Double rank x16	DDR400	DDR400
	Double rank x8	Single rank x8	DDR400	DDR400
	Double rank x8	Single rank x16	DDR400	DDR400
	Double rank x8	Double rank x8	DDR333	DDR400
	Double rank x8	Double rank x16	DDR400	DDR400
	Single rank x16	Single rank x8	DDR400	DDR400
	Single rank x16	Single rank x16	DDR400	DDR400
	Single rank x16	Double rank x8	DDR400	DDR400
	Single rank x16	Double rank x16	DDR400	DDR400
	Double rank x16	Single rank x8	DDR400	DDR400
	Double rank x16	Single rank x16	DDR400	DDR400
	Double rank x16	Double rank x8	DDR400	DDR400
	Double rank x16	Double rank x16	DDR400	DDR400
128-bits	Single rank x8	Single rank x8	DDR400	DDR400
	Double rank x8	Double rank x8	DDR333	DDR400
	Single rank x16	Single rank x16	DDR400	DDR400
	Double rank x16	Double rank x16	DDR400	DDR400

**4.1.3.4 940-pin Lidded Micro PGA Package****Table 48. Registered DIMM Support for 940-pin Lidded Micro PGA Package**

Supported DIMM Configurations	DDR200	DDR266	DDR333	DDR400 <sup>1</sup>
64-bit plus ECC	4	4	4	2
128-bit plus ECC	8	8	8	4
1. See Table 49 for DDR400 speed/loading specifications.				

**Table 49. DDR400 and Quad Rank Registered DIMM Support for 940-pin Lidded Micro PGA Package**

Data Bus	Chip Selects				Maximum DRAM Speed
	H*_DIMM0 H*_MEMCS_L[1:0] & H*_MEMCS_L[5:4]	H*_DIMM1 H*_MEMCS_L[1:0] & H*_MEMCS_L[5:4]	H*_DIMM2 H*_MEMCS_L[3:2] & H*_MEMCS_L[7:6]	H*_DIMM3 H*_MEMCS_L[3:2] & H*_MEMCS_L[7:6]	
64-bits plus ECC	Single rank	N/A	Empty	N/A	DDR400
	Double rank	N/A	Empty	N/A	DDR400
	Empty	N/A	Single rank	N/A	DDR400
	Empty	N/A	Double rank	N/A	DDR400
	Single rank	N/A	Single rank	N/A	DDR400
	Single rank	N/A	Double rank	N/A	DDR400
	Double rank	N/A	Single rank	N/A	DDR400
	Double rank	N/A	Double rank	N/A	DDR333
	Quad rank DDR400	N/A	Empty	N/A	DDR333
	Quad rank < DDR400	N/A	Empty	N/A	DDR266
	Quad rank	N/A	Quad rank	N/A	DDR266
128-bits plus ECC	Single rank	Single rank	Empty	Empty	DDR400
	Double rank	Double rank	Empty	Empty	DDR400
	Empty	Empty	Single rank	Single rank	DDR400
	Empty	Empty	Double rank	Double rank	DDR400
	Single rank	Single rank	Single rank	Single rank	DDR400
	Single rank	Single rank	Double rank	Double rank	DDR400
	Double rank	Double rank	Single rank	Single rank	DDR400
	Double rank	Double rank	Double rank	Double rank	DDR333
	Quad rank DDR400	Quad rank DDR400	Empty	Empty	DDR333
	Quad rank < DDR400	Quad rank < DDR400	Empty	Empty	DDR266
	Quad rank	Quad rank	Quad rank	Quad rank	DDR266

#### 4.1.4 DIMM Matching Algorithm

If DIMMs with different device width, number of rows, number of columns, number of ranks or rank size are used in a 128-bit mode capable system, the following algorithm can be used to maximize utilized system memory for Revision D and earlier revisions.



1. If 128-bit mode is supported in the system and if there is at least one DIMM connected to data bits [127:64], then all DIMMs in the system must be paired. DIMMs are paired if there are two DIMM modules (one connected to data bits [127:64], and one connected to data bits [63:0]) that correspond to the same chip select pair (CS7/6, CS5/4, CS3/2, or CS1/0). If this condition is not satisfied, upper DIMMs should be removed from the DRAM map.
2. Detect the following parameters through the SPD bus: device width, number of rows, number of columns, number of ranks, and rank sizes of each DIMM. A rank is memory selected by a chip select. A DIMM can have one or two ranks.
3. Use the following algorithm to select 64-bit or 128-bit DRAM mode.

```
// Special case
if ((DimmLowSize == 1024MB) && (DimmLowDevWidth == x8) && (DimmLowNRanks == 2) &&
    (DimmHighSize == 512MB) && (DimmHighDevWidth == x16) && (DimmHighNRanks == 1))
    DramMode = 64-bit;
// General case
else{
    if (DimmLowSize <= DimmHighSize){
        DramMode = 128-bit;
    }
    else{
        if (DimmLowSize == DimmHighSize * 2){
            if ((DimmLowNRanks == 1) && (DimmHighNRanks == 2)){
                DramMode = 64-bit;
            }
            else{
                DramMode = 128-bit;
            }
        }
        else{
            DramMode = 64-bit;
        }
    }
}
```

4. If 128-bit mode is selected, find the lowest common denominator for the number of ranks, number of rows, and number of columns and program the memory controller accordingly.

For Revision E and later revisions the Mod64BitMux bit (Function 2: Offset 90h bit 6), can be used to configure the 128-bit interface as two 64-bit interfaces for 939 packages. This mode allows mismatched DIMMs to be installed in the upper and lower DIMM slots without reducing any of the DRAM capacity. When operating in this mode the DRAM interface never operates as a 128-bit interface and the two 64-bit interfaces are not accessed simultaneously.

### 4.1.5 DRAM Initialization

Some DRAM configuration fields must be programmed in a specific order. BIOS must set DramConfigRegLo[8] last to start DRAM initialization after a cold reset not associated with suspend to RAM mode. BIOS must set DramConfigRegLo[12] and DramConfigRegLo[13] last to exit self-

refresh mode after a cold reset associated with suspend to RAM mode. BIOS should not assert LDTSTOP\_L to change HyperTransport™ link width and frequency while DramConfigRegLo[8] or DramConfigRegLo[12] are set.

For Revision C, BIOS must guarantee a delay between setting DramConfigRegHi[25] and setting DramConfigRegLo[12] and DramConfigRegLo[13] when resuming from suspend to RAM mode. The required delay is 110us for registered DIMMs and 10us for unbuffered DIMMs. Both DIMM types require 10us for memory clock stabilization time. Additional 100us required for registered DIMMs is for DIMM PLL stabilization time.

## 4.2 DRAM Configuration

This section shows in a quick reference format how each DRAM controller configuration register should be programmed.

BaseAddrReg[7:0][31:0] = See “DRAM CS Base Address Registers” on page 84.

BaseMaskReg[7:0][31:0] = See “DRAM CS Mask Registers” on page 87.

BankAddrReg[31:0] = See “DRAM Bank Address Mapping Register” on page 88.

DramTimingRegLo[28] = tWR = See “Twr (Write Recovery)” on page 173.

DramTimingRegLo[26:24] = tRP[2:0] = See “tRP (Precharge Command Period)” on page 170.

DramTimingRegLo[23:20] = tRAS[3:0] = See “tRAS (Active-to-Precharge Delay)” on page 170.

DramTimingRegLo[18:16] = tRRD[2:0] = See “tRRD (Active-to-Active of a Different Bank)” on page 171.

DramTimingRegLo[14:12] = tRCD[2:0] = See “tRCD (RAS-to-CAS Delay)” on page 170.

DramTimingRegLo[11:8] = tRFC[3:0] = See “tRFC (Auto-Refresh-to-Active/Auto-Refresh Command Period)” on page 171.

DramTimingRegLo[7:4] = tRC[3:0] = See “tRC (Active-to-Active/Auto-Refresh Command Period)” on page 171.

DramTimingRegLo[2:0] = tCL[2:0] = See “tCL (CAS Latency)” on page 169.

DramTimingRegHi[22:20] = tWCL[2:0] = See “Twcl (Write CAS Latency)” on page 173.

DramTimingRegHi[12:8] = tREF[4:0] = See “tREF (Refresh Rate)” on page 171.

DramTimingRegHi[6:4] = tRWT[2:0] = See “Trwt” on page 173.

DramTimingRegHi[0] = tWTR = See “Twtr (Write to Read Delay)” on page 173.

DramConfigRegLo[27:25] = BypMax[2:0] = 4h

DramConfigRegLo[24] = DisInRcvrs = 0h<sup>1</sup>

DramConfigRegLo[23:20] = x4DIMMs[3:0] = See “x4 DIMMs” on page 172.

DramConfigRegLo[19] = 32ByteEn = 0h<sup>2</sup>

DramConfigRegLo[18] = UnBuffDimm = See “Registered or Unbuffered DIMMs” on page 172.

DramConfigRegLo[17] = DimmEccEn = See “DIMM ECC Enable,”.

DramConfigRegLo[16] = 128/64 = 0h<sup>3</sup>

DramConfigRegLo[15:14] = RdWrQByp = 2h

DramConfigRegLo[13] = SelfRefStat = 0

DramConfigRegLo[12] = ExitSelfRef = 0h

DramConfigRegLo[8] = DramInit = 1h

DramConfigRegLo[2] = QFCEn = 0h

DramConfigRegLo[1] = DrvEn = 0h

DramConfigRegLo[0] = DllDis = 0h

DramConfigRegHi[29:26] = EnMemClk[3:0] = See “Memory Clock Enable” on page 174.

DramConfigRegHi[25] = MemClkRatioVal = 1h

DramConfigRegHi[22:20] = MemClk[2:0] = See “tCL (CAS Latency)” on page 169.

DramConfigRegHi[19] = DynIdleCtrEn = 1h

DramConfigRegHi[18:16] = IdleCycLimit[2:0] = 3h

DramConfigRegHi[11:8] = RdPreamble[3:0] = See “Read Preamble Time” on page 174.

DramConfigRegHo[3:0] = AsyncLat[3:0] = See “Maximum Asynchronous Latency” on page 174.

DramDelayLineLo[31:0] = 0000\_0000h

ScrubControl[31:0] = DramScrubAddrHi[31:0] = DramScrubAddrLo[31:0] = 0000\_0000h

#### **Notes:**

1. In systems that do not use the DRAM controller, it is preferable to disable DRAM input receivers and output drivers and to leave input receivers unconnected. This bit should be set in such systems.
2. This bit should be set for 64-bit interfaces (when DramConfigRegLo[16] is cleared) on a platform with a graphics core (AGP or UMA) that issues 32-byte requests. This bit is ignored for 128-bit interfaces (when DramConfigRegLo[16] is set).
3. This bit should be cleared for 64-bit interfaces and set for 128-bit interfaces. The value of this bit is a function of the system design and cannot be determined via the SPD ROM.



## 5 Machine Check Architecture

---

The AMD Athlon™ 64 processor and AMD Opteron™ processor machine check mechanism allows the processor to detect and report a variety of hardware (or machine) errors found when reading and writing data, probing, cache-line fills and writebacks. These include parity errors associated with caches and TLBs, ECC errors associated with caches and DRAM, as well as system bus errors associated with reading and writing to the external bus.

Software can enable the processor to report machine check errors through the machine check exception (See “#MC—Machine Check Exception” in *AMD64 Architecture Programmer's Manual*, Volume 2: System Programming). Most machine check exceptions do not allow reliable restarting of the interrupted programs. However, error conditions are logged in a set of model-specific registers (MSRs) that can be used by system software to determine the possible source of a hardware problem.

### 5.1 Determining Machine Check Support

The availability of machine check registers and support of the machine check exception is implementation dependent. System software executes the CPUID instruction to determine whether a processor implements these features. After CPUID is executed, the values of the machine check architecture (MCA) bit and the machine check exception (MCE) bit loaded in the EDX register indicate whether the processor implements the machine check registers and the machine check exception, respectively. See “Processor Feature Identification” in *AMD64 Architecture Programmer's Manual*, Volume 2: System Programming, and “CPUID” in *AMD64 Architecture Programmer's Manual*, Volume 3: General Purpose and System Instructions, for further information on the level of machine check support.

Once system software determines that the machine check registers are available, it must determine the extent of processor support for the machine check mechanism. This is accomplished by reading the machine check capabilities register (MCG\_CAP). See “Machine Check Global Capabilities Register” in *AMD64 Architecture Programmer's Manual*, Volume 2: System Programming, for more information on the interpretation of the MCG\_CAP contents.

### 5.2 Machine Check Errors

Machine check errors are either recoverable or irrecoverable. Recoverable errors are those that the processor can correct and, thus, do not raise the machine check exception (#MC). However, the error is still logged in the machine check MSRs and it is the responsibility of the system software to periodically poll the machine check MSRs to determine if recoverable errors have occurred. If a recoverable error has been logged in the machine check MSRs, a second recoverable error can overwrite it.

Irrecoverable, or fatal, machine check errors cannot be corrected by the processor. If machine check is enabled, they raise the machine check exception (#MC). If an irrecoverable error has been logged in the machine check MSRs, a second recoverable or irrecoverable MCA error will not overwrite it but will set a bit that indicates overflow.

In the case of both recoverable and irrecoverable MCA errors, the contents of the machine check MSRs are maintained through a warm reset. This is helpful since in some cases it may not be possible to invoke the machine check handler due to the error and this allows the BIOS or other system boot software to recover and report the information associated with the error.

## 5.2.1 Sources of Machine Check Errors

The processor can detect errors from the following hardware blocks within the processor. Each block forms an error reporting bank for the purpose of reporting machine check errors.

- Data cache unit (DC)—Includes the cache structures that hold data and tags, the data TLBs, and the data cache probing logic.
- Instruction cache unit (IC)—Includes the instruction cache structures that hold instructions and tags, the instruction TLBs, and the instruction cache probing logic.
- Bus unit (BU)—Includes the system bus interface to the Northbridge and the level 2 cache.
- Load/store unit (LS)—Includes logic used to manage loads and stores.
- Northbridge unit (NB)—Includes the Northbridge and DRAM controller.

A scrubber is associated with the data cache in DC, the L2 cache tag array in bus unit, and the DRAM in the Northbridge. A scrubber is a hardware widget that periodically wakes up during idle cache cycles and inspects the next line of the array with which it is associated to look for errors. If it finds a single bit ECC error, the scrubber corrects the error and prevents a regular access from encountering the same error. This is important in the data cache, since all ECC errors encountered during regular operation in the data cache are fatal. In the bus unit and the Northbridge, this scrubber function also saves the additional time it would take to fix single-bit ECC errors when they are encountered during normal operation.

Even though 64-bits are read from the data cache during normal operation, a byte load will use one byte, a word load will use two bytes, and a double word load will use 4 bytes. If a data cache ECC error is detected in a byte that is not being used by a load, it is corrected like an error detected by the data cache scrubber. This feature is referred to as "piggyback scrubbing".

Table 50 on page 191 shows the various sources of machine check errors that can be encountered in the processor. ECC check on the linefill data for IC data is done by the DC unit. However, the ECC error is reported by IC which is the unit responsible for receiving the data. The data cache is protected by ECC; however single bit ECC errors encountered by normal load and store accesses are not corrected. When the data cache scrubber encounters a single-bit ECC error, it corrects it.

**Table 50. Sources of Machine Check Errors**

Unit	Error Source	Type of Check	Recoverable
DC	System line fill into the data cache	ECC	Yes—single bit error
	L2 cache line fill into the data cache		No—multiple bit error
	Cache data—data array	ECC	Yes—if single bit ECC error is detected by the scrubber No—any other case different than single bit ECC error detected by the scrubber
	Cache data—snoop tag array	Parity	No
	Cache data—tag array		No
	L1 Data TLB—physical and virtual arrays		No
	L2 Data TLB—physical and virtual arrays		No
IC	System line fill into the instruction cache	ECC	Yes—single bit error
	L2 cache line fill into the instruction cache		No—multiple bit error
	Instruction cache—data array	Parity	Yes <sup>1</sup>
	Instruction cache—tag array		Yes <sup>1</sup>
	Instruction cache—snoop tag array		No
	L1 Instruction TLB—physical and virtual arrays		Yes <sup>1</sup>
	L2 Instruction TLB—physical and virtual arrays		Yes <sup>1</sup>
	System address out of range		No, but detection is precise
BU	L2 cache data array	ECC	Yes—single bit error
	L2 cache—tag array (during scrubs)		No—multiple bit error
	L2 cache—tag array	Parity	No. Single and multiple bit errors in an L2 cache tag can also be detected, but not corrected.
	System Address Out-of-Range	Read Data	No
LS	System Address Out-of-Range	Read Data	No. Loads are detected precisely and stores are detected imprecisely.
NB	See “Machine Check Architecture (MCA) Registers” on page 121 for information on Northbridge machine check errors.		
<b>Notes:</b> 1. Instruction cache lines are invalidated and refetched.			

## 5.3 Machine Check Architecture Registers

The processor architecture defines a set of model-specific registers (MSRs) to support the MCA mechanism. They are used to configure the MCA functions and provide a way for the hardware to report errors in a manner compatible with the machine check architecture. These registers include the global MCA registers used to set up machine checks and additional banks of MSRs for recording errors that are detected by the hardware blocks listed in “Sources of Machine Check Errors” on page 190. They can be read and written using the RDMSR and WRMSR instructions. The following is a complete list of MCA MSRs.

- Global status and control registers
  - Machine check capabilities MSR (MCG\_CAP)
  - Processor status MSR (MCG\_STATUS)
  - Exception reporting control MSR (MCG\_CTL)
- Each error reporting bank (associated with a specific hardware block listed in “Sources of Machine Check Errors” on page 190) contains the following registers:
  - Error reporting control register (MCi\_CTL)
  - Error reporting control register mask (MCi\_CTL\_MASK)
  - Error reporting status register (MCi\_STATUS)
  - Error reporting address register (MCi\_ADDR)
  - Machine check miscellaneous error information register (MCi\_MISC)

The *i* in each register name corresponds to the number of a supported register bank. Each error-reporting register bank is associated with a specific processor unit (or group of processor units). The number of error-reporting register banks is implementation-specific.

Software reads the MCG\_CAP register to determine the number of supported register banks. The AMD Athlon™ 64 and AMD Opteron™ Processors support five banks. The first error-reporting register (MC0\_CTL) always starts with MSR address 400h, followed by MC0\_STATUS (401h), MC0\_ADDR (402h), and MC0\_MISC (403h). Error-reporting-register MSR addresses are assigned sequentially through the remaining supported register banks. Using this information, software can access all error-reporting registers in an implementation-independent manner.

The global machine check registers as well as the generic form of each register in the error reporting banks are now described. The specifics of each error reporting bank will be described in “Error Reporting Banks” on page 199.

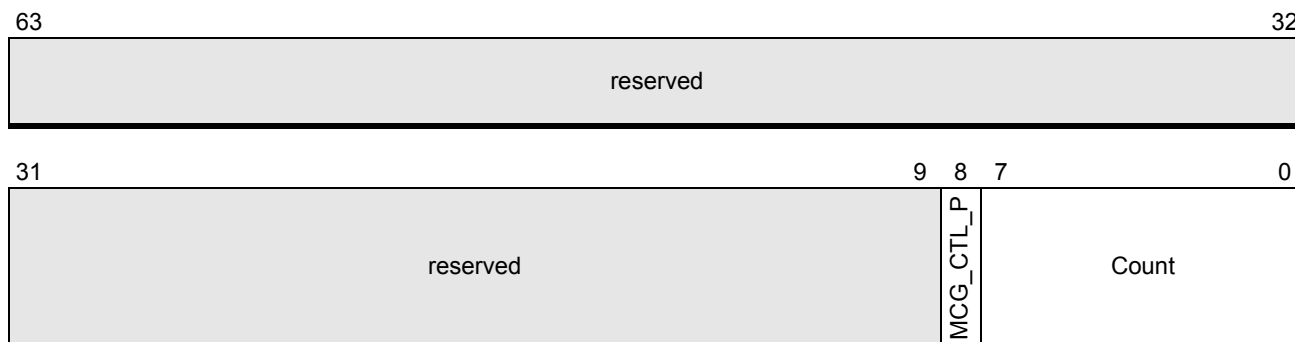
### 5.3.1 Global Machine Check Model-Specific Registers (MSRs)

The global MSRs supported by the machine check mechanism include the MCG\_CAP, the MCG\_STATUS and the MCG\_CTL registers.



**5.3.1.1 MCG\_CAP—Global Machine Check Capabilities Register**

This read-only register indicates the capabilities of the processor machine check architecture. Attempting to write to this register will result in a #GP exception. See *AMD64 Architecture Programmer's Manual, Volume 2: System Programming*, for more details.

**MCG\_CAP Register****MSR 0179h**

Bit	Name	Function	R/W	Reset
63–9	reserved			0
8	MCG_CTL_P	MCG_CTL Register Present	R	1
7–0	Count	Count	R	05h

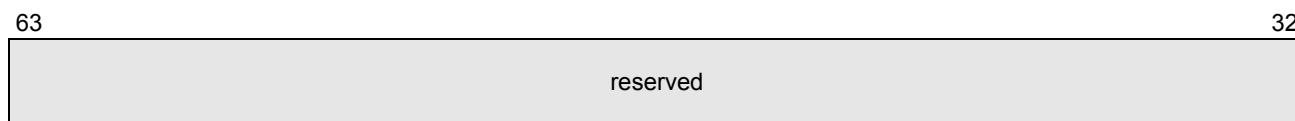
**Field Descriptions**

**Count (Count)**—Bits 7–0. Number of error-reporting banks supported by the processor implementation.

**MCG\_CTL Register Present (MCG\_CTL\_P)**—Bit 8. Indicates if the MCG\_CTL register is present. When the bit is set to 1, the register is supported. When the bit is cleared to 0, the register is unsupported.

**5.3.1.2 MCG\_STATUS—Global Machine Check Processor Status Register**

This register contains basic information about the processor state after a machine check error is detected. See *AMD64 Architecture Programmer's Manual, Volume 2: System Programming*, for more details.

**MCG\_STATUS Register****MSR 017Ah**

[illegible]

Bit	Name	Function	R/W	Reset
63–3	reserved			0
2	MCIP	Machine Check in Progress	R/W	0
1	EIPV	Error IP Valid Flag	R/W	0
0	RIPV	Restart IP Valid Flag	R/W	0

## Field Descriptions

**Restart IP Valid Flag (RIPV)**—Bit 0. When set, indicates if program execution can be restarted at EIP pushed on stack.

**Error IP Valid Flag (EIPV)—**Bit 1. When set, indicates if the EIP pushed on stack is that of the instruction which caused the detection of the machine check error.

**Machine Check in Progress (MCIP)**—Bit 2. When set, indicates that a machine check is in progress.

### 5.3.1.3 MCG\_CTL—Global Machine Check Exception Reporting Control Register

This register contains a global bit for each error-checking unit in the processor. Each bit enables the reporting, through the MCA interface, of errors detected by that particular unit. See *AMD64 Architecture Programmer's Manual*, Volume 2: System Programming, for more details.

## MCG\_CTL Register

MSR 017Bh

Diagram illustrating a reserved field structure. The field is divided into two sections: a 63-bit section on the left and a 32-bit section on the right, both of which are reserved.

31																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Bit	Name	Function	R/W	Reset
63–5	reserved			0
4	NBE	NB Register Bank Enable	R/W	0
3	LSE	LS Register Bank Enable	R/W	0
2	BUE	BU Register Bank Enable	R/W	0
1	ICE	IC Register Bank Enable	R/W	0
0	DCE	DC Register Bank Enable	R/W	0

## Field Descriptions

**DC Register Bank Enable (DCE)**—Bit 0. When set, indicates that the Data Cache register bank is enabled.

**IC Register Bank Enable (ICE)**—Bit 1. When set, indicates that the Instruction Cache register bank is enabled.

**BU Register Bank Enable (BUE)**—Bit 2. When set, indicates that the Bus Unit register bank is enabled.

**LS Register Bank Enable (LSE)**—Bit 3. When set, indicates that the Load/Store register bank is enabled.

**NB Register Bank Enable (NBE)**—Bit 4. When set, indicates that the Northbridge register bank is enabled.

## 5.3.2 Error Reporting Bank Machine Check MSRs

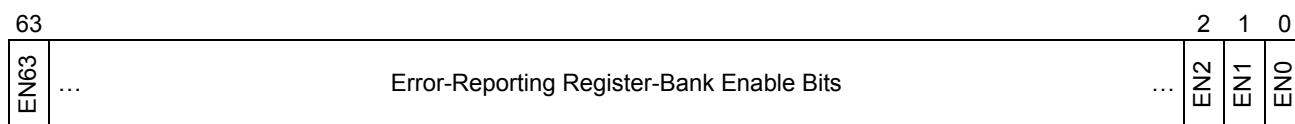
The registers in each error reporting bank include `MCi_CTL`, `MCi_CTL_MASK`, `MCi_STATUS` and `MCi_ADDR`. The `MCi_MISC` register is not supported in AMD Athlon™ 64 and AMD Opteron™ Processors.

### 5.3.2.1 `MCi_CTL`—Machine Check Control Registers

The machine check control registers (`MCi_CTL`) contain an enable bit for each error source within an error-reporting register bank. Setting an enable bit to 1 enables error-reporting for the specific feature controlled by the bit, and clearing the bit to 0 disables error reporting for the feature. These registers should generally be set to either all zeros or all ones. As described in Section 5.4.2.2, each enable bit can be masked by the `MCi_CTL_MASK` register.

#### `MCi_CTL` Registers

**MSRs 0400h, 0404h, 0408h, 040Ch, 0410h**



Bit	Name	Function	R/W	Reset
63–0	EN63–EN0	Enables	R/W	0

## Field Descriptions

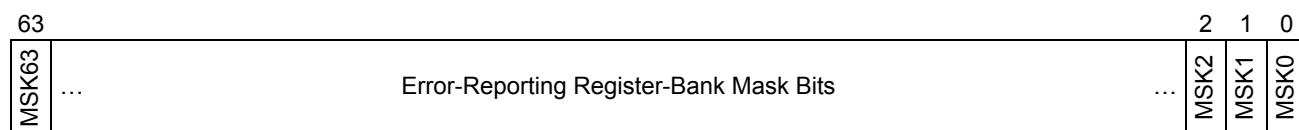
**Enables (EN63–EN0)**—Bits 63–0. If set, error reporting for the specific feature controlled by the bit is enabled. Not all these bits may be implemented in each bank.

### 5.3.2.2 MCi\_CTL\_MASK—Machine Check Control Mask Registers

The machine check control register masks (MCi\_CTL\_MASK) provide another level of control in enabling and disabling specific error reporting features. Each bit set to 1 in the MCi\_CTL\_MASK register will inhibit the setting of the corresponding enable bit in the MCi\_CTL register that it is associated with. This register is typically programmed by BIOS and not by the Kernel software.

#### MCi\_CTL\_MASK Registers

MSR C001\_0044h, C001\_0045h, C001\_0046h,  
C001\_0047h, C001\_0048h



Bit	Name	Function	R/W	Reset
63–0	MSK63–MSK0	Control Register Masks	R/W	0

#### Field Descriptions

**Control Register Masks (MSK63–MSK0)**—Bits 63–0. If set, the corresponding bit in the MCi\_CTL register will be forced to 0 when written.

0 = Masking not enabled

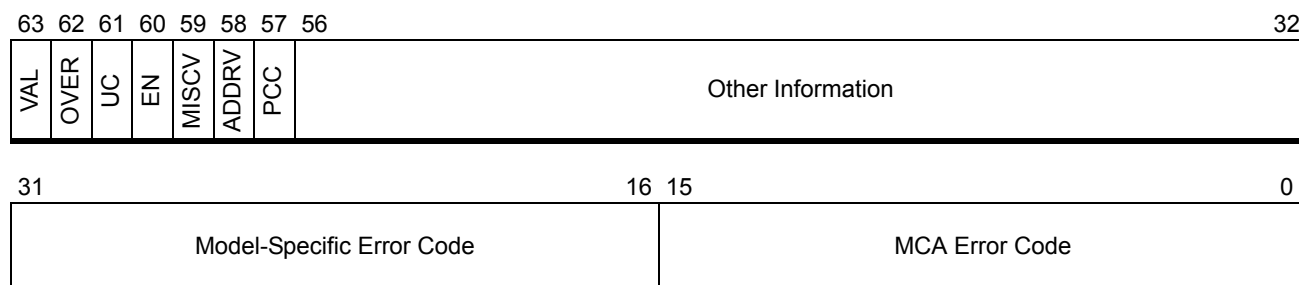
1 = Masking enabled

### 5.3.2.3 MCi\_STATUS—Machine Check Status Registers

The machine check status register (MCi\_STATUS) describes the error detected by its unit. It is written by the processor and should be cleared to 0 by software; writing any other value to the register causes a general protection (GP#) exception. When a machine check error occurs, the processor loads an error code into bits [15:0] of the appropriate MCi\_STATUS register. Errors are classified as either system bus, cache, or TLB errors. All the status registers in each bank use the same general logging format shown below.

#### MCi\_STATUS Registers

MSRs 0401h, 0405h, 0409h, 040Dh, 0411h



Bit	Name	Function	R/W	Reset
63	VAL	Valid	R/W	X
62	OVER	Status Register Overflow	R/W	X
61	UC	Uncorrected Error indication	R/W	X
60	EN	Error Condition Enabled	R/W	X
59	MISCV	Miscellaneous-Error Register Valid	R/W	X
58	ADDRV	Error-Address Register Valid	R/W	X
57	PCC	Processor-Context Corrupt	R/W	X
56–32		Other Information	R/W	X
31–16		Model-Specific Error Code	R/W	X
15–0		MCA Error Code	R/W	X

## Field Descriptions

**MCA Error Code**—Bits 15–0. This field holds an error code when an error is detected.

**Model-Specific Error Code**—Bits 31–16. This field encodes model-specific information about the error.

**Other Information**—Bits 56–32. This field holds model-specific error information.

**Processor-Context Corrupt (PCC)**—Bit 57. If set to 1, this bit indicates that the state of the processor may be corrupted by the error condition. Reliable restarting might not be possible.  
0 = Processor not corrupted  
1 = Processor may be corrupted

**Error-Address Register Valid (ADDRV)**—Bit 58. If set to 1, this bit indicates that the address saved in the address register is the address where the error occurred.  
0 = Address register not valid  
1 = Address register valid

**Miscellaneous-Error Register Valid (MISCV)**—Bit 59. If set to 1, this bit indicates whether the Miscellaneous Error register contains valid information for this error. This bit should always be 0 since the Miscellaneous Error register is not implemented.

**Error Condition Enabled (EN)**—Bit 60. If set to 1, this bit indicates that MCA error reporting is enabled for this error in the MCA Control register.  
0 = Error checking not enabled  
1 = Error checking enabled

**Uncorrected Error indication (UC)**—Bit 61. If set to 1, this bit indicates that the error was not corrected by hardware.  
0 = Error corrected  
1 = Error not corrected

**Status Register Overflow (OVER)**—Bit 62. Set to 1 if the unit detects an error but the valid bit of this register already set. Enabled errors are written over disabled errors, uncorrectable errors are written over correctable errors. Uncorrectable errors are not overwritten.

0 = No error overflow

1 = Error overflow

**Valid (VAL)**—Bit 63. If set to 1, this bit indicates that a valid error has been detected by the unit. This bit should be cleared to 0 by software after the register is read.

0 = No valid error detected

1 = Valid error detected

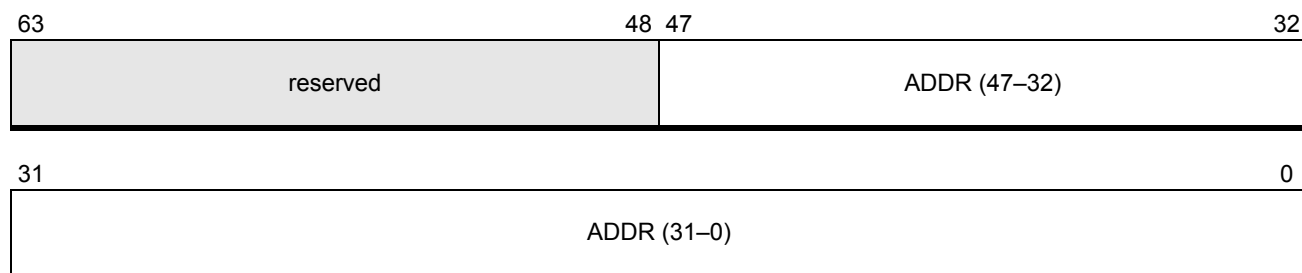
#### 5.3.2.4 MCi\_ADDR—Machine Check Address Registers

Each error-reporting register bank includes a machine check address register (MCi\_ADDR) that the processor uses to report the instruction memory address or data memory address responsible for the machine check error. The contents of this register are valid only if the ADDR\_V bit in the corresponding MCi\_STATUS register is set to 1.

The address can be up to 48 bits wide. In reality, not all address bits may be valid and the address field of the MCi\_ADDR varies in width. The address field can hold either a virtual (linear) or physical address, depending on the type of error. Some error types cause the processor to load valid values into a subset of the address bits. The bit ranges depend on the values in the MCi\_STATUS register, i.e., on the type of error detected by the unit, which can be determined by examining the MCA error code contained in the MCi\_STATUS register.

#### MCi\_ADDR Registers

MSRs 0402h, 0406h, 040Ah, 040Eh, 0412h



Bit	Name	Function	R/W	Reset
63–48	reserved			0
47–0	ADDR	Address	R/W	0

#### Field Descriptions

**Address (ADDR)**—Bits 47–0. Not all these bits may be valid. The valid bits depend on the type of error registered in the corresponding MCi\_STATUS register.

## 5.4 Error Reporting Banks

The AMD Athlon™ 64 and AMD Opteron™ Processors have five error-reporting banks—DC, IC, BU, LS, and NB. Each error reporting bank includes the following registers:

- Machine check control register (MCi\_CTL).
- Error reporting control register mask (MCi\_CTL\_MASK).
- Machine check status register (MCi\_STATUS).
- Machine check address register (MCi\_ADDR).

The general format of these registers was described in “Error Reporting Bank Machine Check MSRs” on page 195. This section describes the specifics of each register as it relates to each error reporting bank.

### 5.4.1 Data Cache (DC)

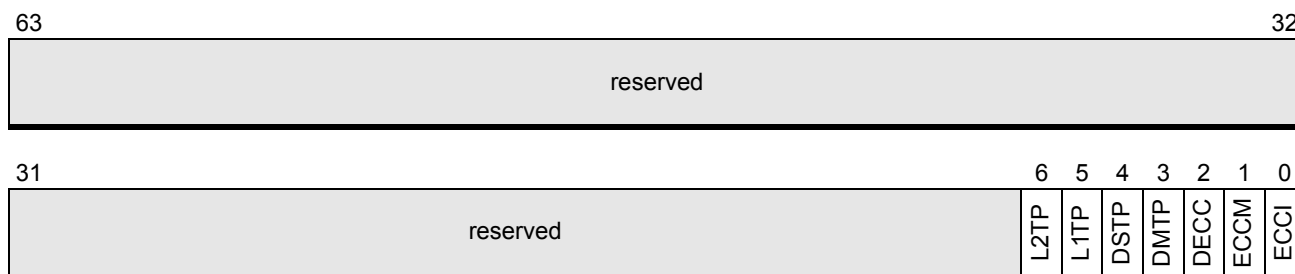
The data cache unit includes the level 1 data cache that holds data and tags, as well as two levels of TLBs and data cache probing logic.

#### 5.4.1.1 MC0\_CTL—DC Machine Check Control Register

This register enables the reporting, via the MCA interface, of a variety of errors detected by the Data Cache (DC) processor unit. For an error to be reported, both the global DC enable, MCG\_CTL[DCE], and the corresponding local enable shown below must be set. If the local enable is off, the error will only be logged in MC0\_STATUS, but not reported via the machine check exception. If the global DC enable is off, no error will be logged or reported.

#### MC0\_CTL Register

MSR 0400h



Bit	Name	Function	R/W	Reset
63–7	reserved			0
6	L2TP	L2 TLB Parity Errors	R/W	0
5	L1TP	L1 TLB Parity Errors	R/W	0
4	DSTP	Snoop Tag Array Parity Errors	R/W	0
3	DMTP	Main Tag Array Parity Errors	R/W	0

Bit	Name	Function	R/W	Reset
2	DECC	Data Array ECC Errors	R/W	0
1	ECCM	Multi-bit ECC Data Errors	R/W	0
0	ECCI	Single-bit ECC Data Errors	R/W	0

## Field Descriptions

**Single-bit ECC Data Errors (ECCI)**—Bit 0. Report single-bit ECC data errors during data cache line fills or TLB reloads from the internal L2 or the system.

**Multi-bit ECC Data Errors (ECCM)**—Bit 1. Report multi-bit ECC data errors during data cache line fills or TLB reloads from the internal L2 or the system.

**Data Array ECC Errors (DECC)**—Bit 2. Report data cache data array ECC errors.

**Main Tag Array Parity Errors (DMTP)**—Bit 3. Report data cache main tag array parity errors.

**Snoop Tag Array Parity Errors (DSTP)**—Bit 4. Report data cache snoop tag array parity errors.

**L1 TLB Parity Errors (L1TP)**—Bit 5. Report data cache L1 TLB parity errors.

**L2 TLB Parity Errors (L2TP)**—Bit 6. Report data cache L2 TLB parity errors.

### 5.4.1.2 MC0\_CTL\_MASK—DC Machine Check Control Mask Register

The MC0\_CTL\_MASK register, at address MSR C001\_0044h, can be used to mask the enabling of individual error reporting features in DC. Any bit set to 1 in this register will inhibit writing 1s to the corresponding bits of the MC0\_CTL register. See “MCi\_CTL\_MASK—Machine Check Control Mask Registers” on page 196 for a description of this register.

### 5.4.1.3 MC0\_STATUS—DC Machine Check Status Register

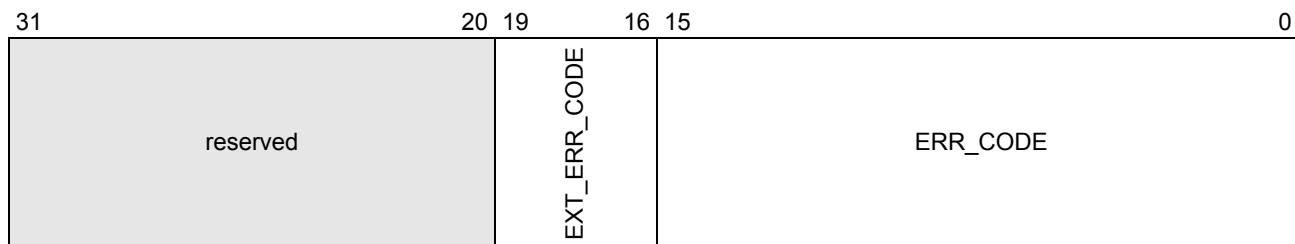
The MC0\_STATUS MSR describes the error that was detected by DC. The machine check mechanism writes the status register bits when an error is detected and sets the register valid bit (bit 63) to 1 to indicate that the status information is valid. This register will be written even if error reporting for the detected error is not enabled. However, error reporting must be enabled for the error to result in a machine check exception. Software is responsible for clearing this register.

## MC0\_STATUS Register

**MSR 0401h**

63	62	61	60	59	58	57	56	55	54		47	46	45	44		41	40	39		32
VAL	OVER	UC	EN	MISCV	ADDRV	PCC	reserved	SYND				CECC	UECC	reserved			SCRUB	reserved		





Bit	Name	Function	R/W	Reset
63	VAL	Valid.	R/W	X
62	OVER	Second error detected	R/W	X
61	UC	Error not corrected	R/W	X
60	EN	Error reporting enabled	R/W	X
59	MISCV	Additional info in MCi_MISC	R/W	X
58	ADDRV	Error address in MCi_ADDR	R/W	X
57	PCC	Processor state corrupted by error	R/W	X
56–55	reserved			0
54–47	SYND	ECC Syndrome (7–0)	R/W	X
46	CECC	Correctable ECC error	R/W	X
45	UECC	Uncorrectable ECC error	R/W	X
44–41	reserved		R/W	0
40	SCRUB	Error detected on a scrub	R/W	X
39–20	reserved			0
19–16	EXT_ERR_CODE	Extended error code	R/W	X
15–0	ERR_CODE	Error subsection	R/W	X

## Field Descriptions

**Error Subsection (ERR\_CODE)**—Bits 15–0. Indicates in which subsection the error was detected and what transaction initiated it. See Table 18 on page 128 for the format of the error code and Tables 19 through 24 on pages 128–129 for descriptions of the subfields.

**Extended Error Code (EXT\_ERR\_CODE)**—Bits 19–16. Contains an extended error code.

*DC/IC:*

0000b = TLB parity error in physical array

0001b = TLB parity error in virtual array (multi-match error)

*BU:*

0000b = Bus or cache data array error

0010b = Cache tag array error

*LS:* Reserved

**Table 51. DC Error Codes**

Error Type	Extended Error Code	Error Code						
		Type	PP	T	RRRR	II	LL	TT
System Line Fill	0000	BUS	SRC	0	DRD	MEM/IO	LG	-
L2 Cache Line Fill	0000	Memory	-	-	DRD	-	L2	Data
Data Load/ Store/ Victim/ Snoop	0000	Memory	-	-	DRD/ DWD/ Evict/ Snoop	-	L1	Data
Data Scrub	0000	Memory	-	-	GEN	-	L1	Data
Tag Snoop/ Victim	0000	Memory	-	-	Snoop/ Evict	-	L1	Data
Tag Load/ Store	0000	Memory	-	-	DRD/ DWD	-	L1	Data
L1 TLB	0000	TLB	-	-	-	-	L1	Data
L1 TLB Multimatch	0001	TLB	-	-	-	-	L1	Data
L2 TLB	0000	TLB	-	-	-	-	L2	Data
L2 TLB Multimatch	0001	TLB	-	-	-	-	L2	Data

**Error Detected on a Scrub (SCRUB)—Bit 40.**

*DC:* If set, indicates that the error was detected on a scrub.

*IC/BU/LS:* Reserved.

**Uncorrectable ECC Error (UECC)—Bit 45.** If set, indicates that the error is an uncorrectable ECC error.

**Correctable ECC Error (CECC)—Bit 46.** If set, indicates that the error is a correctable ECC error.

**ECC Syndrome Bits 7–0 (SYND)—Bits 54–47.**

*DC:* The lower 8 syndrome bits when an ECC error is detected.

*IC/BU/LS:* Reserved.

**Processor State Corrupted By Error (PCC)—Bit 57.** If set, indicates that the processor state may have been corrupted by the error condition.

**Error address in MCI\_ADDR (ADDRV)—Bit 58.** If set, indicates that the address saved in the corresponding MCI\_ADDR register is the address where the error occurred.

**Additional Info in MCi\_MISC (MISCV)**—Bit 59. If set, indicates that the MCi\_MISC register contains additional info. This bit is always set to 0 on an AMD Athlon™ 64 or AMD Opteron™ processor.

**Error Reporting Enabled (EN)**—Bit 60. If set, indicates that error reporting was enabled for this error in the corresponding MCi\_CTL register.

**Error Not Corrected (UC)**—61. If set, it indicates that the error was not corrected.

**Second Error Detected (OVER)**—Bit 62. Set if a second error is detected while the VAL bit of this register is already set.

**Valid (VAL)**—Bit 63. Set if the information in this register is valid.

**Table 52. DC Error Status Bit Settings**

Error Type	UC	ADDRV	PCC	ECC_Synd Valid	CECC	UECC	SCRUB
System Line Fill	If multi-bit	1	If multi-bit	Y	If single-bit	If multi-bit	0
L2 Cache Line Fill	If multi-bit	1	If multi-bit	Y	If single-bit	If multi-bit	0
Data Load/Store/Victim/Snoop	1	1/0	1	Y	If single-bit	If multi-bit	0
Data Scrub	If multi-bit	1	0	Y	If single-bit	If multi-bit	1
Tag Snoop/Victim	1	1/0	1	N	0	0	0
Tag Load/Store	1	1	1	N	0	0	0
L1 TLB	1	1	1	N	0	0	0
L1 TLB Multimatch	1	1	1	N	0	0	0
L2 TLB	1	1	1	N	0	0	0
L2 TLB Multimatch	1	1	1	N	0		0

#### 5.4.1.4 MC0\_ADDR—DC Machine Check Address Register (MSR 0402h)

The contents of this register are valid only if the ADDRv bit in the MC0\_STATUS register is set to 1. As shown in “MCi\_ADDR—Machine Check Address Registers” on page 198, the address can be up to 48 bits wide. In reality, not all address bits may be valid and the address field of the MC0\_ADDR

varies in width. The bit ranges depend on the values in the MC0\_STATUS register (i.e., the type of error detected) as shown in Table 53.

**Table 53. Valid MC0\_ADDR Bits**

Error Source	Access Type	Valid Address Bits
Snoop Tag Array	snoop	physical[11–6]
	victim	
Data Tag Array	load	physical[39–3]
	store	
Data Array	victim	physical[11–6]
	snoop	
	load	physical[39–3]
	store	
	scrub	physical[11–3]
L1 Data TLB	load, store	linear[47–12]
L2 Data TLB		
L2 Cache Data	line-fill	physical[39–6]
System Data		

## 5.4.2 Instruction Cache (IC)

The IC unit includes the level 1 instruction cache that holds instruction data and tags, as well as two levels of TLBs and instruction cache probing logic.

### 5.4.2.1 MC1\_CTL—IC Machine Check Control Register

This register enables the reporting, via the MCA interface, of a variety of errors detected by the Instruction Cache (IC) processor unit. For an error to be reported, both the global IC enable, MCG\_CTL[ICE], and the corresponding local enable shown below must be set. If the local enable is off, the error will only be logged in MC1\_STATUS, but not reported via the machine check exception. If the global IC enable is off, no error will be logged or reported.

#### MC1\_CTL Register

**MSR 0404h**

63	reserved															32					
31	reserved										10	9	8	7	6	5	4	3	2	1	0
										RDDE	reserved		L2TP	L1TP	ISTP	IMTP	IDP	ECCM	ECCI		

Bit	Name	Function	R/W	Reset
63–10	reserved			0
9	RDDE	Read Data Errors	R/W	0
8–7	reserved			0
6	L2TP	L2 TLB Parity Errors	R/W	0
5	L1TP	L1 TLB Parity Errors	R/W	0
4	ISTP	Snoop tag array parity errors	R/W	0
3	IMTP	Main tag array parity errors	R/W	0
2	IDP	Data array parity errors	R/W	0
1	ECCM	Multi-bit ECC data errors	R/W	0
0	ECCI	Single-bit ECC data errors	R/W	0

## Field Descriptions

**Single-bit ECC Data Errors (ECCI)**—Bit 0. Report single-bit ECC data errors during instruction cache line fills or TLB reloads from the internal L2 or the system.

**Multi-bit ECC Data Errors (ECCM)**—Bit 1. Report multi-bit ECC data errors during instruction cache line fills or TLB reloads from the internal L2 or the system.

**Data Array Parity Errors (IDP)**—Bit 2. Report instruction cache data array parity errors.

**Main Tag Array Parity Errors (IMTP)**—Bit 3. Report instruction cache main tag array parity errors.

**Snoop Tag Array Parity Errors (ISTP)**—Bit 4. Report instruction cache snoop tag array parity errors.

**L1 TLB Parity Errors (L1TP)**—Bit 5. Report instruction cache L1 TLB parity errors.

**L2 TLB Parity Errors (L2TP)**—Bit 6. Report instruction cache L2 TLB parity errors.

**Read Data Errors (RDDE)**—Bit 9. Report system read data errors for an instruction cache fetch if MC2\_CTL[S\_RDE\_ALL] = 1.

### 5.4.2.2 MC1\_CTL\_MASK—IC Machine Check Control Mask Register (MSR C001\_0045h)

The MC1\_CTL\_MASK register can be used to mask the enabling of individual error reporting features in IC. Any bit set to 1 in this register will inhibit writing 1s to the corresponding bits of the MC1\_CTL register. See “MCi\_CTL\_MASK—Machine Check Control Mask Registers” on page 196 for a description of this register.

**5.4.2.3 MC1\_STATUS—IC Machine Check Status Register (MSR 0405h)**

The MC1\_STATUS MSR describes the error that was detected by IC and is very similar to MC0\_STATUS. See “MC0\_STATUS—DC Machine Check Status Register” on page 200 for a description of the fields in this register

**Table 54. IC Error Codes**

Error Type	Extended Error Code	Error Code						
		Type	PP	T	RRRR	II	LL	TT
System Line Fill	0000	BUS	SRC	0	IRD	MEM	LG	-
L2 Cache Line Fill	0000	Memory	-	-	IRD	-	L2	Instruction
IC Data Load	0000	Memory	-	-	IRD	-	L1	Instruction
Tag Snoop/ Victim	0000	Memory	-	-	Snoop/ Evict	-	L1	Instruction
Tag Load/ Victim	0000	Memory	-	-	IRD/ Evict	-	L1	Instruction
L1 TLB	0000	TLB	-	-	-	-	L1	Instruction
L1 TLB Multimatch	0001	TLB	-	-	-	-	L1	Instruction
L2 TLB	0000	TLB	-	-	-	-	L2	Instruction
L2 TLB Multimatch	0001	TLB	-	-	-	-	L2	Instruction
System Data Read Error	0000	BUS	SRC	0	IRD	MEM	LG	-

**Table 55. IC Error Status Bit Settings**

Error Type	UC	ADDRV	PCC	Syndrome	CECC	UECC	SCRUB
System Line Fill	If multi-bit	1	If multi-bit	N	If single-bit	If multi-bit	0
L2 Cache Line Fill	If multi-bit	1	If multi-bit	N	If single-bit	If multi-bit	0
IC Data Load	0	1	0	N	0	0	0
Tag Snoop/ Victim	1	1/0	1	N	0	0	0
Tag Load/ Victim	0	1/0	0	N	0	0	0

**Table 55. IC Error Status Bit Settings (Continued)**

Error Type	UC	ADDRV	PCC	Syndrome	CECC	UECC	SCRUB
L1 TLB	0	1	0	N	0	0	0
L1 TLB Multimatch	0	1	0	N	0	0	0
L2 TLB	0	1	0	N	0	0	0
L2 TLB Multimatch	0	1	0	N	0		0
System Data Read Error	1	0	0	N	0	0	0

**5.4.2.4 MC1\_ADDR—IC Machine Check Address Register (MSR 0406h)**

The contents of this register are valid only if the ADDRv bit in the MC1\_STATUS register is set to 1. The address can be up to 48 bits wide. In reality, not all address bits may be valid and the address field of the MC1\_ADDR varies in width. The bits ranges depend on the values in the MC1\_STATUS register (i.e., the type of error detected) and this is shown in Table 56.

**Table 56. Valid MC1\_ADDR Bits**

Error Source	Access Type	Valid Address Bits
Snoop Tag Array	Snoop	Physical[39–6]
	Victim	None
Instruction Tag Array	Code Read	Linear[47–6]
	Victim	None
Instruction Data Array	Code Read	Linear[47–4]
L1 TLB	Code Read	Linear[47–12] for 4-Kbyte page Linear[47–20] for 2-Mbyte page
L2 TLB	Code Read	Linear[47–12] for 4-Kbyte page
L2 Cache Data	Line-fill	Physical[39–6]
System Data		
System Address Out of Range		None

**5.4.3 Bus Unit (BU)**

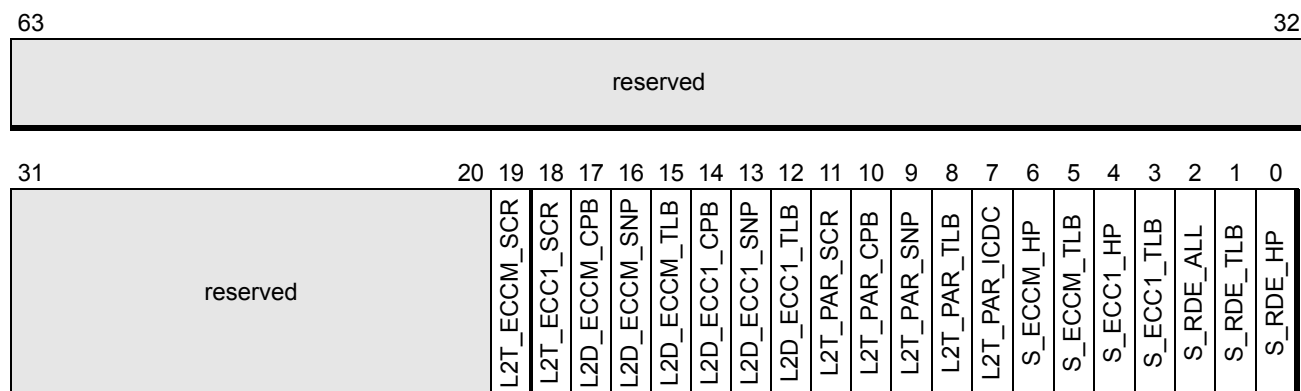
The bus unit consists of the system bus interface logic and the L2 cache.

**5.4.3.1 MC2\_CTL—BU Machine Check Control Register**

This register enables the reporting, via the MCA interface, of a variety of errors detected in the Bus processor unit (BU). For an error to be reported, both the global BU enable, MCG\_CTL[BUE], and

the corresponding local enable shown below must be set. If the local enable is off, the error will only be logged in MC2\_STATUS, but not reported via the machine check exception. If the global BU enable is off, no error will be logged or reported.

## MC2\_CTL Register

**MSR 0408h**


Bit	Name	Function	R/W	Reset
63–20	reserved			0
19	L2T_ECCM_SCR	L2 tag array multi-bit ECC Scrub	R/W	0
18	L2T_ECC1_SCR	L2 tag array 1-bit ECC Scrub	R/W	0
17	L2D_ECCM_CPB	L2 data array multi-bit ECC copyback	R/W	0
16	L2D_ECCM_SNP	L2 data array multi-bit ECC snoop	R/W	0
15	L2D_ECCM_TLB	L2 data array multi-bit ECC TLB reload	R/W	0
14	L2D_ECC1_CPB	L2 data array 1-bit ECC copyback	R/W	0
13	L2D_ECC1_SNP	L2 data array 1-bit ECC snoop	R/W	0
12	L2D_ECC1_TLB	L2 data array 1-bit ECC TLB reload	R/W	0
11	L2T_PAR_SCR	L2 tag array parity scrub	R/W	0
10	L2T_PAR_CPB	L2 tag array parity copyback	R/W	0
9	L2T_PAR_SNP	L2 tag array parity snoop	R/W	0
8	L2T_PAR_TLB	L2 tag array parity TLB reload	R/W	0
7	L2T_PAR_ICDC	L2 tag array parity IC or DC fetch	R/W	0
6	S_ECCM_HP	System data multi-bit ECC hardware prefetch	R/W	0
5	S_ECCM_TLB	System data multi-bit ECC TLB reload	R/W	0
4	S_ECC1_HP	System data 1-bit ECC hardware prefetch	R/W	0
3	S_ECC1_TLB	System data 1-bit ECC TLB reload	R/W	0
2	S_RDE_ALL	All system read data	R/W	0
1	S_RDE_TLB	System read data TLB reload	R/W	0
0	S_RDE_HP	System read data hardware prefetch	R/W	0

## Field Descriptions

**System Read Data Hardware Prefetch (S\_RDE\_HP)**—Bit 0. Report system read data errors for a hardware prefetch.



**System Read Data TLB Reload (S\_RDE\_TLB)**—Bit 1. Report system read data errors for a TLB reload.

**All System Read Data (S\_RDE\_ALL)**—Bit 2. Report system read data errors for any operation including a DC/IC fetch, TLB reload or hardware prefetch.

**System Data 1-bit ECC TLB Reload (S\_ECC1\_TLB)**—Bit 3. Report system data 1-bit ECC errors for a TLB reload.

**System Data 1-bit ECC Hardware Prefetch (S\_ECC1\_HP)**—Bit 4. Report system data 1-bit ECC errors for a hardware prefetch.

**System Data Multi-bit ECC TLB Reload (S\_ECCM\_TLB)**—Bit 5. Report system data multi-bit ECC errors for a TLB reload.

**System Data Multi-bit ECC Hardware Prefetch (S\_ECCM\_HP)**—Bit 6. Report system data multi-bit ECC errors for a hardware prefetch.

**L2 Tag Array Parity IC or DC Fetch (L2T\_PAR\_ICDC)**—Bit 7. Report L2 tag array parity errors for an IC or DC fetch.

**L2 Tag Array Parity TLB Reload (L2T\_PAR\_TLB)**—Bit 8. Report L2 tag array parity errors for a TLB reload.

**L2 Tag Array Parity Snoop (L2T\_PAR\_SNP)**—Bit 9. Report L2 tag array parity errors for a snoop.

**L2 Tag Array Parity Copyback (L2T\_PAR\_CPB)**—Bit 10. Report L2 tag array parity errors for a copyback.

**L2 Tag Array Parity Scrub (L2T\_PAR\_SCR)**—Bit 11. Report L2 tag array parity errors for a scrub.

**L2 Data Array 1-bit ECC TLB Reload (L2D\_ECC1\_TLB)**—Bit 12. Report L2 data array 1-bit ECC errors for a TLB reload.

**L2 Data Array 1-bit ECC Snoop (L2D\_ECC1\_SNP)**—Bit 13. Report L2 data array 1-bit ECC errors for a snoop.

**L2 Data Array 1-bit ECC Copyback (L2D\_ECC1\_CPB)**—Bit 14. Report L2 data array 1-bit ECC errors for a copyback.

**L2 Data Array Multi-bit ECC TLB Reload (L2D\_ECCM\_TLB)**—Bit 15. Report L2 data array multi-bit ECC errors for a TLB reload.

**L2 Data Array Multi-bit ECC Snoop (L2D\_ECCM\_SNP)**—Bit 16. Report L2 data array multi-bit ECC errors for a snoop.

**L2 Data Array Multi-bit ECC Copyback (L2D\_ECCM\_CPB)**—Bit 17. Report L2 data array multi-bit ECC errors for a copyback.

**L2 Tag Array 1-bit ECC Scrub (L2T\_ECC1\_SCR)**—Bit 18. Report L2 tag array 1-bit ECC errors for a scrub.

**L2 Tag Array Multi-bit ECC Scrub (L2T\_ECCM\_SCR)**—Bit 19. Report L2 tag array multi-bit ECC errors for a scrub.

### 5.4.3.2 MC2\_CTL\_MASK—BU Machine Check Control Mask Register (MSR C001\_0046h)

The MC2\_CTL\_MASK register can be used to mask the enabling of individual error reporting features in BU. Any bit set to 1 in this register will inhibit writing 1s to the corresponding bits of the MC2\_CTL register. See “MCi\_CTL\_MASK—Machine Check Control Mask Registers” on page 196 for a description of this register.

### 5.4.3.3 MC2\_STATUS—BU Machine Check Status Register (MSR 0409h)

The MC2\_STATUS MSR describes the error that was detected by BU and is very similar to MC0\_STATUS. See “MC0\_STATUS—DC Machine Check Status Register” on page 200 for a description of the fields in this register.

**Table 57. BU Error Codes**

Error Type	Access Type	Extended Error Code	Error Code						
			Type	PP	T	RRRR	II	LL	TT
Tag Parity	Instruction Fetch	0010	Memory	-	-	IRD	-	L2	Instruction
	Data Fetch	0010	Memory	-	-	DRD	-	L2	Data
	TLB/Snoop/Evict	0010	Memory	-	-	RD/Snoop/Evict	-	L2	Generic
	Scrub	0010	Memory	-	-	GEN	-	L2	Generic
Tag ECC	Scrub	0010	Memory	-	-	GEN	-	L2	Instruction
System Data Read Error	TLB	0000	BUS	SRC	0	RD	MEM	LG	-
	Hardware Prefetch	0000	BUS	SRC	0	Prefetch	MEM/IO	LG	-
System Data ECC	TLB	0000	BUS	SRC	0	RD	MEM/IO	LG	-
	Hardware Prefetch	0000	BUS	SRC	0	Prefetch	MEM	LG	-
Data ECC	TLB	0000	Memory	-	-	RD	-	L2	Generic
Data Copyback	Snoop/Evict	0000	Memory	-	-	Snoop/Evict	-	L2	Generic

**Table 58. BU Error Status Bit Settings**

Error Type	Access Type	UC	ADDRV	PCC	Syndrome	CECC	UECC	SCRUB
Tag Parity	Instruction Fetch	1	1	1	N	0	0	0
	Data Fetch	1	1	1	N	0	0	0
	TLB/ Snoop/ Evict	1	1	1	N	0	0	0
	Scrub	1	1	0	N	0	0	1
Tag ECC	Scrub	If multi-bit	1	0	Y	If single-bit	If multi-bit	1
System Data Read Error	TLB	1	1	0	N	0	0	0
	Hardware Prefetch	1	0	0	N	0	0	0
System Data ECC	TLB	If multi-bit	1	If multi-bit	Y	If single-bit	If multi-bit	0
	Hardware Prefetch	If multi-bit	0	If multi-bit	Y	If single-bit	If multi-bit	0
Data ECC	TLB	If multi-bit	0	If multi-bit	Y	If single-bit	If multi-bit	0
Data Copyback	Snoop/ Evict	If multi-bit	0	If multi-bit	Y	If single-bit	If multi-bit	0

#### 5.4.3.4 MC2\_ADDR—BU Machine Check Address Register (MSR 040Ah)

The contents of this register are valid only if the ADDR\_V bit in the MC2\_STATUS register is set to 1. The address can be up to 48 bits wide. In reality, not all address bits may be valid and the address field of the MC2\_ADDR varies in width. The bit ranges depend on the values in the MC2\_STATUS register (i.e., the type of error detected) and this is shown in Table 59.

**Table 59. Valid MC2\_ADDR Bits**

Error Source	Access Type	Valid Address Bits
L2 Cache—Data Array	Any	Physical[39–6]
L2 Cache—Tag Array	Any	MC2_ADDR[3–0] contains encoded cache way Physical[15–6] for 1-Mbyte L2 Physical[14–6] for 512-Kbyte L2 Physical[13–6] for 256-Kbyte L2 Physical[12–6] for 128-Kbyte L2
System Address Out of Range	Any	Physical[39–6]

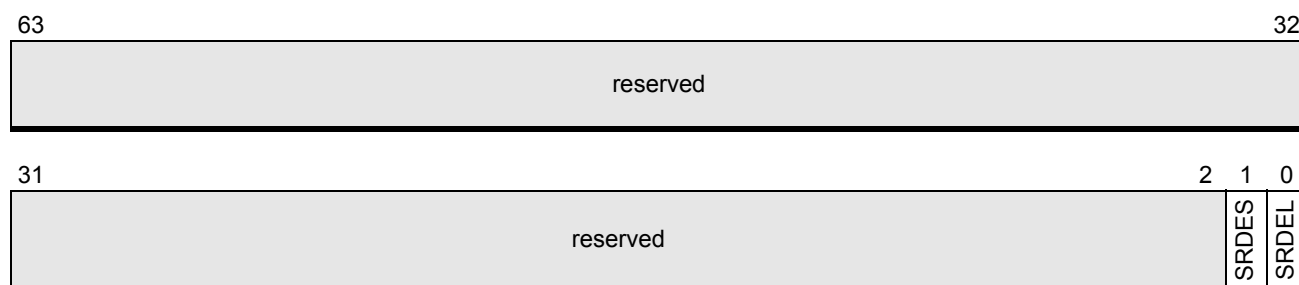
## 5.4.4 Load Store Unit (LS)

### 5.4.4.1 MC3\_CTL—LS Machine Check Control Register

This register enables the reporting, via the MCA interface, of two types of errors detected by the Load/Store (LS) processor unit. For an error to be reported, both the global LS enable, MCG\_CTL[LSE], and the corresponding local enable shown below must be set. If the local enable is off, the error will only be logged in MC3\_STATUS, but not reported via the machine check exception. If the global LS enable is off, no error will be logged or reported.

#### MC3\_CTL Register

MSR 040Ch



Bit	Name	Function	R/W	Reset
63–2	reserved			0
1	S_RDE_S	Read Data Errors on Store	R/W	0
0	S_RDE_L	Read Data Errors on Load	R/W	0

#### Field Descriptions

**Read Data Errors on Load (S\_RDE\_L)**—Bit 0. Report system read data errors on a load if MC2\_CTL[S\_RDE\_ALL] = 1.

**Read Data Errors on Store (S\_RDE\_S)**—Bit 1. Report system read data errors on a store if MC2\_CTL[S\_RDE\_ALL] = 1.

### 5.4.4.2 MC3\_CTL\_MASK—LS Machine Check Control Mask Register (MSR C001\_0047h)

The MC3\_CTL\_MASK register can be used to mask the enabling of individual error reporting features in LS. Any bit set to 1 in this register will inhibit writing 1s to the corresponding bits of the MC3\_CTL register. See “MCi\_CTL\_MASK—Machine Check Control Mask Registers” on page 196 for a description of this register.

#### 5.4.4.3 MC3\_STATUS—LS Machine Check Status Register (MSR 040Dh)

The MC3\_STATUS MSR describes the error that was detected by LS and is very similar to MC0\_STATUS. See “MC0\_STATUS—DC Machine Check Status Register” on page 200 for a description of the fields in this register.

#### 5.4.4.4 MC3\_ADDR—LS Machine Check Address Register (MSR 040Eh)

The contents of this register are valid only if the ADDR\_V bit in the MC3\_STATUS register is set to 1. The only type of error recorded by the LS machine check mechanism is a “system address out of range” or read data error for which MC3\_ADDR[39:0] store the physical address.

### 5.4.5 Northbridge (NB)

The Northbridge and DRAM memory controller are included in this block.

#### 5.4.5.1 MC4\_CTL—NB Machine Check Control Register (MSR 0410h)

This register enables the reporting, via the MCA interface, of the errors detected by the North Bridge (NB) processor unit. “MCA NB Control Register” (Function 3, Offset 40h) maps to MC4\_CTL[31:0]. See “MCA NB Control Register” on page 121 for detailed information on this register. For an error to be reported, both the global NB enable, MCG\_CTL[NBE], and the corresponding local enable must be set. If the local enable is off, the error will only be logged in MC4\_STATUS, but not reported via the machine check exception. If the global NB enable is off, no error will be logged or reported.

#### 5.4.5.2 MC4\_CTL\_MASK—NB Machine Check Control Mask Register (MSR C001\_0048h)

The MC4\_CTL\_MASK register can be used to mask the enabling of individual error reporting features in the Northbridge. Any bit set to 1 in this register will inhibit writing 1s to the corresponding bits of the MC4\_CTL register. See “MCi\_CTL\_MASK—Machine Check Control Mask Registers” on page 196 for a description of this register.

#### 5.4.5.3 MC4\_STATUS—NB Machine Check Status Register (MSR 0411h)

The MC4\_STATUS MSR describes the error that was detected by the Northbridge. “MCA NB Status Low Register” (Function 3, Offset 48h) maps to MC4\_STATUS MSR[31:0] and “MCA NB Status High Register” (Function 3, Offset 4Ch) maps to MC4\_STATUS MSR[63:32]. See “MCA NB Status Low Register” on page 127 and “MCA NB Status High Register” on page 130 for more detailed information.

#### 5.4.5.4 MC4\_ADDR—NB Machine Check Address Register (MSR 0412h)

The contents of this register are valid only if the ErrAddrVal bit in the MC4\_STATUS register is set to 1. “MCA NB Address Low Register” (Function 3, Offset 50h) maps to MC4\_ADDR[31:0] and

“MCA NB Address High Register” (Function 3, Offset 54h) maps to MC4\_ADDR[63:32]. See “MCA NB Address Low Register” on page 133 and “MCA NB Address High Register” on page 133 for a description of this register.

## 5.5 Initializing the Machine Check Mechanism

Following is the general process system software should follow to initialize the machine check mechanism:

1. Execute the CPUID and verify that the processor supports the machine check exception (MCE) and MCA. MCE is supported when EDX bit 7 is set to 1, and MCA is supported when EDX bit 14 set to 1. Software should not proceed with initializing the machine check mechanism if MCE is not supported.
2. If MCA is supported, system software should take the following steps:

- a. Check to see if the MCG\_CTL\_P bit in the MCG\_CAP register is set to 1. If it is, then the MCG\_CTL register is supported by the processor. When this register is supported, software should set its enable bits to 1 for the machine check features it uses. Software can load MCG\_CTL with all 1s to enable all machine check features.
- b. Read the COUNT field from the MCG\_CAP register to determine the number of error-reporting register banks supported by the processor. This is set to 5 in AMD Athlon™ 64 and AMD Opteron™ Processors since there are five blocks—DC, IC, BU, LS, and NB. For each error-reporting register bank, software should set the enable bits to 1 in the MCi\_CTL register for the error types it wants the processor to report. Software can load each MCi\_CTL register bit with a 1 to enable all error-reporting mechanisms.

The error-reporting register banks are numbered from 0 to one less than the value found in the MCG\_CAP.COUNT field. For example, when the COUNT field indicates that 5 register banks are supported, they are numbered 0 to 4.

- c. For each error-reporting register bank, software should clear all status fields in the MCi\_STATUS register by writing all 0s to the register.

It is possible that valid error status is reported in the MCi\_STATUS registers at the time software clears them. The status can reflect fatal errors recorded before a processor reset or errors recorded during the system power-up and boot process. Prior to clearing the MCi\_STATUS registers, software should examine their contents and log any errors found.

3. As a final step in the initialization process, system software should enable the machine check exception by setting CR4.MCE (bit 6) to 1.

## 5.6 Using Machine Check Features

System software can detect and handle machine check errors using two methods:

- Software can periodically examine the machine check status registers for reported errors, and log any errors found.
- Software can enable the machine check exception (#MC). When an uncorrectable error occurs, the processor immediately transfers control to the machine check exception handler. In this case, system software provides a machine check exception handler that, at a minimum, logs detected errors. The exception handler can be designed for a specific processor implementation or can be generalized to work on multiple implementations.

### 5.6.1 Handling Machine Check Exceptions

The processor uses the interrupt control-transfer mechanism to invoke an exception handler after a machine check exception occurs. This requires system software to initialize the interrupt-descriptor table (IDT) with either an interrupt gate or a trap gate that references the interrupt handler.

At a minimum, the machine check exception handler must be capable of logging errors for later examination. Because most machine check errors are not recoverable, the ability to log errors can be sufficient for implementation of the handler. More thorough exception-handler implementations can analyze errors to determine if each error is recoverable. If a recoverable error is identified, the exception handler can attempt to correct the error and restart the interrupted program.

Machine check exception handlers that attempt to correct recoverable errors must be thorough in their analysis and the corrective actions they take. The following guidelines should be used when writing such a handler:

- All status registers in the error-reporting register banks must be examined to identify the cause or causes of the machine check exception. Read the COUNT field from MCG\_CAP to determine the number of status registers supported by the processor. The status registers are numbered from 0 to one less than the value found in the MCG\_CAP.COUNT field. For example, if the COUNT field indicates five status registers are supported, they are numbered MC0\_STATUS to MC4\_STATUS.
- Check the valid bit in each status register (MCi\_STATUS.VAL). The MCi\_STATUS register does not need to be examined when its valid bit is clear.
- Check the valid MCi\_STATUS registers to see if error recovery is possible. Error recovery is not possible when:
  - The processor-context corrupt bit (MCi\_STATUS.PCC) is set to 1.
  - The error-overflow status bit (MCi\_STATUS.OVER) is set to 1. This bit indicates that more than one machine check error has occurred, but only one error is reported by the status register.

If error recovery is not possible, the handler should log the error information and return to the operating system.

- Check the MCi\_STATUS.UC bit to see if the processor corrected the error. If UC=1, the processor did not correct the error, and the exception handler must correct the error prior to restarting the interrupted program. If the handler cannot correct the error, it should log the error information and return to the operating system.

- When identifying the error condition, portable exception handlers should examine only the MCA error-code field of the MCi\_STATUS register. See the error codes in tables 19 through 24 for information on interpreting this field.

If the MCG\_STATUS.RIPV bit is set to 1, the interrupted program can be restarted reliably at the instruction-pointer address pushed onto the exception-handler stack. If RIPV = 0, the interrupted program cannot be restarted reliably, although it may be possible to restart it for debugging purposes.

- When logging errors, particularly those that are not recoverable, check the MCG\_STATUS.EIPV bit to see if the instruction-pointer address pushed onto the exception-handler stack is related to the machine check error. If EIPV = 0, the address is not guaranteed to be related to the error.
- Prior to exiting the machine check handler, be sure to clear MCG\_STATUS.MCIP to 0. MCIP indicates that a machine check exception occurred. If this bit is set when another machine check exception occurs, the processor enters the shutdown state.
- When an exception handler is able to, at a minimum, successfully log an error condition, clear the MCi\_STATUS registers prior to exiting the machine check handler. Software is responsible for clearing at least the MCi\_STATUS.VAL bit.
- Additional machine check exception-handler portability can be added by having the handler use the CPUID instruction to identify the processor and its capabilities. Implementation-specific software can be added to the machine check exception handler based on the processor information reported by CPUID.

### 5.6.1.1 Reporting Correctable Machine Check Errors

Machine check exceptions do not occur if the error is correctable by the processor. If system software wishes to log and report correctable machine check errors, a system-service routine must be provided to check the contents of the machine check status registers for correctable errors. The service routine can be invoked by system software on a periodic basis, or it can be manually invoked by the user as needed.

Assuming that the processor supports the machine check registers, a service routine that reports correctable errors should perform the following:

1. Examine each status register (MCi\_STATUS) in the error-reporting register banks. For each MCi\_STATUS register with a set valid bit (VAL = 1), the service routine should:
  - a. Save the contents of the MCi\_STATUS register.
  - b. Save the contents of the corresponding MCi\_ADDR register if MCi\_STATUS.ADDRV = 1.
  - c. Save the contents of the corresponding MCi\_MISC register if MCi\_STATUS.MISCV = 1.
  - d. Check to see if MCG\_STATUS.MCIP = 1, indicating that the machine check exception handler is in progress. If this is the case, then the machine check exception handler has called the service routine to log the errors. In this situation, the error-logging service routine should determine whether or not the interrupted program is restartable, and report the determination back to the exception handler. The program is *not restartable* if either of the following is true:



- `MCi_STATUS.PCC = 1`, indicating the processor context is corrupted.
  - `MCG_STATUS.RIPV = 0`, indicating the interrupted program cannot be restarted reliably at the instruction-pointer address pushed onto the exception-handler stack.
2. Once the information found in the error-reporting register banks is saved, the `MCi_STATUS` register should be cleared to 0. This allows the processor to properly report any subsequent errors in the `MCi_STATUS` registers.
  3. In multiprocessor configurations, the service routine can save the processor node identifier. This can help locate a failing multiprocessor-system component, which can then be isolated from the rest of the system.



## 6 System Management Mode (SMM)

---

System management mode (SMM) is an operating mode entered when system management interrupt SMI is asserted. The SMI is handled by a dedicated interrupt handler pointed to by processor model specific registers. SMM is used for system control activities such as power management. These activities are transparent to conventional operating systems (such as MS-DOS and Windows® operating system). SMM is used by the BIOS and specialized low level device drivers. The code and data for SMM are stored in the SMM memory area, which may be isolated from the main memory accesses using special processor functions.

This chapter describes the SMM state save area, entry into and exit from SMM, exceptions and interrupts in SMM, and memory allocation and addressing in SMM.

### 6.1 SMM Overview

The processor enters SMM after the system logic asserts the SMI interrupt and the processor recognizes SMI active. The processor may be programmed to send a special bus cycle to the system, indicating that it is entering SMM mode. The processor saves its state into the SMM memory state save area and jumps to the SMM service routine. The processor returns from SMM by executing the RSM instruction in the SMM service routine. The processor restores its state from the SMM state save area and resumes execution of the instruction following the point where it entered SMM. The processor may be programmed to send a special bus cycle to the system, indicating that it is exiting SMM mode.

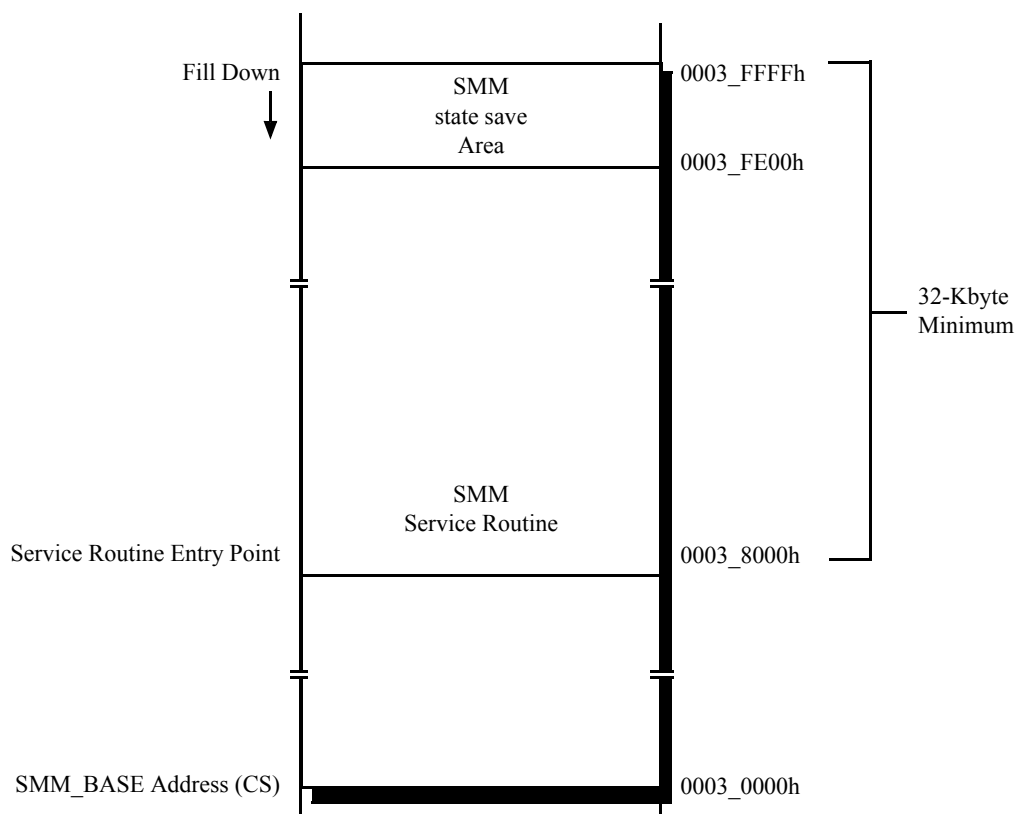
### 6.2 Operating Mode and Default Register Values

The software environment after entering SMM mode has the following characteristics:

- Addressing and operation in Real mode.
- 4-Gbyte segment limits.
- Default 16-bit operand, address, and stack sizes (instruction prefixes can override these defaults).
- Control transfers that do not override the default operand size truncate the EIP to 16 bits.
- Far jumps or calls cannot transfer control to a segment with a base address requiring more than 20 bits, like in Real mode segment-base addressing, unless a change is made into protected mode
- A20M# is disabled. A20M# assertion or deassertion have no effect on SMM mode.
- Interrupt vectors use the Real mode interrupt vector table.
- The IF flag in EFLAGS is cleared (INTR is not recognized).

- The TF flag in EFLAGS is cleared.
- The NMI and INIT interrupts are masked.
- Debug register DR7 is cleared (debug traps are disabled).

Figure 4 shows the default map of the SMM memory area. It consists of a 64-Kbyte area, between 0003\_0000h and 0003\_FFFFh. The top 32 Kbytes (0003\_8000–0003\_FFFFh) must be populated with RAM. The default code-segment (CS) base address for the area (called the SMM\_BASE address) is 0003\_0000h. The top 512 bytes (0003\_FE00–0003\_FFFFh) are the SMM state save area. The default entry point for the SMM service routine is 0003\_8000h.



**Figure 4. Default SMM Memory Map**

## 6.3 SMM State Save Definition

**Table 60. SMM Save State (Offset FE00–FFFFh)**

Offset (Hex) from SMM_BASE <sup>1</sup>	Contents		Size	Allowable Access
FE00h	ES	Selector	Word	Read-Only
FE02h		Attributes	Word	
FE04h		Limit	Doubleword	
FE08h		Base	Quadword	
FE10h	CS	Selector	Word	Read-Only
FE12h		Attributes	Word	
FE14h		Limit	Doubleword	
FE18h		Base	Quadword	
FE20h	SS	Selector	Word	Read-Only
FE22h		Attributes	Word	
FE24h		Limit	Doubleword	
FE28h		Base	Quadword	
FE30h	DS	Selector	Word	Read-Only
FE32h		Attributes	Word	
FE34h		Limit	Doubleword	
FE38h		Base	Quadword	
FE40h	FS	Selector	Word	Read-Only
FE42h		Attributes	Word	
FE44h		Limit	Doubleword	
FE48h		Base	Quadword	
FE50h	GS	Selector	Word	Read-Only
FE52h		Attributes	Word	
FE54h		Limit	Doubleword	
FE58h		Base	Quadword	
FE60h–FE61h	GDTR	reserved	2 Bytes	Read-Only
FE62h		reserved	Word	
FE64h		Limit	Word	
FE66h–FE67h		reserved	2 Bytes	
FE68h		Base	Quadword	
<b>Notes:</b> 1. The offset for the SMM-revision identifier is compatible with previous implementations.				

**Table 60. SMM Save State (Offset FE00–FFFFh) (Continued)**

Offset (Hex) from SMM_BASE <sup>1</sup>	Contents		Size	Allowable Access
FE70h	LDTR	Selector	Word	Read-Only
FE72h		Attributes	Word	
FE74h		Limit	Doubleword	
FE78h		Base	Quadword	
FE80h–FEB1h	IDTR	reserved	2 Bytes	Read-Only
FE82h		reserved	Word	
FE84h		Limit	Word	
FEB6h–FEB7h		reserved	2 Bytes	
FE88h		Base	Quadword	
FE90h	TR	Selector	Word	Read-Only
FE92h		Attributes	Word	
FE94h		Limit	Doubleword	
FE98h		Base	Quadword	
FEA0h–FEBFh	reserved		32 Bytes	—
FEC0h–FEC3h	SMM I/O Trap		Doubleword	Read-Only
FEC4h–FEC7h	reserved		4 Bytes	—
FEC8h	I/O Instruction Restart		Byte	Read/Write
FEC9h	Auto-Halt Restart		Byte	
FECAh	NMI Mask		Byte	
FECBh–FECFh	reserved		6 Bytes	—
FED0h	EFER		Quadword	Read-Only
FED8h–FEFBh	reserved		36 Bytes	—
FEFCh	SMM-Revision Identifier <sup>1</sup>		Doubleword	Read-Only
FF00h	SMM_BASE		Doubleword	Read/Write
FF04h–FF47h	reserved		68 Bytes	—
FF48h	CR4		Quadword	Read-Only
FF50h	CR3		Quadword	
FF58h	CR0		Quadword	
FF60h	DR7		Quadword	Read-Only
FF68h	DR6		Quadword	
FF70h	RFLAGS		Quadword	Read/Write
<b>Notes:</b>				
1. The offset for the SMM-revision identifier is compatible with previous implementations.				

**Table 60. SMM Save State (Offset FE00–FFFFh) (Continued)**

Offset (Hex) from SMM_BASE <sup>1</sup>	Contents	Size	Allowable Access
FF78h	RIP	Quadword	Read/Write
FF80h	R15	Quadword	
FF88h	R14	Quadword	
FF90h	R13	Quadword	
FF98h	R12	Quadword	
FFA0h	R11	Quadword	
FFA8h	R10	Quadword	
FFB0h	R9	Quadword	
FFB8h	R8	Quadword	
FFC0h	RDI	Quadword	Read/Write
FFC8h	RSI	Quadword	
FFD0h	RBP	Quadword	
FFD8h	RSP	Quadword	
FFE0h	RBX	Quadword	
FFE8h	RDX	Quadword	
FFF0h	RCX	Quadword	
FFF8h	RAX	Quadword	
<b>Notes:</b>			
1. The offset for the SMM-revision identifier is compatible with previous implementations.			

After recognizing the SMI assertion, the processor saves almost the entire integer processor state to memory. Exceptions are: the floating point state, the model specific registers, and CR2. Any processor state not saved in the save state must be saved and restored by the SMM handler. With the AMD Athlon™ 64 processor and AMD Opteron™ processor extension of register size to 64-bits, the SMM save state has changed considerably compared to the legacy 32-bit version. One exception is the SMM Revision Identifier, which is located in the same position as before, for proper SMM identification.

**Note:** The save state will always be 64-bit, regardless of the operating mode (32-bit or 64-bit).

## 6.4 SMM Initial State

After storing the save state, the processor starts executing the handler at address SMM\_BASE + 08000h. It starts with the entry state listed in Table 61.

**Table 61. SMM Entry State**

Register	Initial Contents		
	Selector	Base	Limit
CS	SMM_BASE[19:4]	SMM_BASE[31:0]	4 Gbytes
DS	0000h	0000_0000h	4 Gbytes
ES	0000h	0000_0000h	4 Gbytes
FS	0000h	0000_0000h	4 Gbytes
GS	0000h	0000_0000h	4 Gbytes
SS	0000h	0000_0000h	4 Gbytes
General-Purpose Registers	Unmodified		
EFLAGS	0000_0002h		
EIP	0000_8000h		
CR0	Bits 0, 2, 3, and 31 cleared (PE, EM, TS, and PG); remainder is unmodified		
CR4	0000_0000h		
GDTR	Unmodified		
LDTR	Unmodified		
IDTR	Unmodified		
TR	Unmodified		
DR7	0000_0400h		
DR6	Undefined		

## 6.5 SMM-Revision Identifier

The SMM-Revision Identifier specifies the version of SMM and the processor extensions.

### SMM-Revision Identifier

Offset FEFCh

31	18	17	16	15	0
reserved				BRL IOTRAP	REV

Bit	Name	Function	R/W	Reset
31–18	reserved	SBZ	R	
17	BRL	SMM Base Relocation Supported	R	1
16	IOTRAP	SMM I/O Trap Supported	R	1
15–0	REV	SMM Revision Level	R	0064h



## 6.6 SMM Base Address

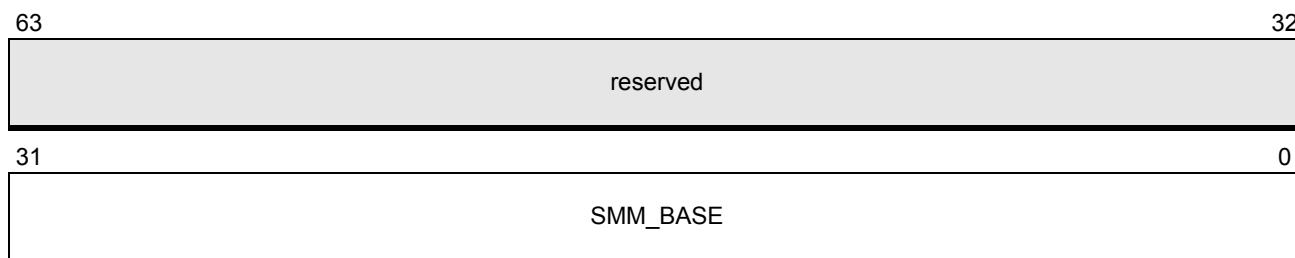
SMM\_BASE register holds the base of the system management memory region, and its default value is 30000h. The value of this register is stored in the save state on entry into SMM and it is restored on returning from SMM. The first SMI handler instruction is fetched at SMM\_BASE + 8000h. The 16 bit code segment selector is formed on entering SMM from SMM\_BASE[19–4]. SMM\_BASE[31–20] and SMM\_BASE[3–0] have to be 0; if not, the CS base will not be equal to the (CS selector << 4), and attempts to load the CS\_BASE into other descriptors will not work properly, since the selector cannot properly represent the base. If there is a far branch in the SMM handler, it will only be able to address the lower 1M of memory and will not be able to restore the SMM base to a value above 1M, unless it first switches to protected mode.

The SMM base address can be changed in two ways:

- The SMM base address, at offset FF00h in the SMM state save area, can be changed by the SMM service routine. The RSM instruction will update the SMM\_BASE with the new value.
- The SMM\_BASE is mapped to MSR C001\_0111h and WRMSR instruction can be used to update it.

### SMM\_BASE Address

Offset FF00h  
MSR C001\_0111h



Bit	Name	Function	R/W
63–32	reserved	RAZ	
31–0	SMM_BASE	System management mode base	R/W

## 6.7 Auto Halt Restart

During entry into SMM, the auto halt restart byte at offset FEC9h in the SMM state save area indicates whether SMM was entered from the Halt state. Before returning from SMM, the halt restart byte bit can be written by the SMM service routine to specify whether the return from SMM should take the processor back to the Halt state or to the instruction-execution state specified by the SMM state save area (the instruction after the HLT).

If the return from SMM takes the processor back to the Halt state, the HLT instruction is not refetched and reexecuted, but the Halt special bus cycle is sent to the system on the bus and the processor enters the HLT state.

## Auto Halt Restart

Offset FEC9h



Bit	Name	Function	R/W
7–1	reserved	SBZ	R/W
0	HLT	HLT Restart Byte	R/W

## Field Description

**HLT Restart Byte (HLT)**—Bit 0. On entry:

- 0 = Entered from normal x86 instruction boundary
- 1 = Entered from HLT State

After entry and before returning:

- 0 = Return to SMM State
- 1 = Return to HLT State

## 6.8 SMM I/O Trap and I/O Restart

If the assertion of SMI is recognized on the boundary of an I/O instruction, the I/O trap doubleword at offset FEC0h in the SMM state save area contains information about the executed I/O instruction. If the Valid bit is equal to 0, not all information is valid.

This is used in the I/O restart implementation. When an I/O access is done to a device that may be unavailable, the system can assert SMI and trap the I/O instruction. The system can then determine which device is not responding by using the I/O port information in the I/O Trap doubleword. It can then figure out why the device is not responding, fix the device, and re-execute the I/O instruction. It can reconstruct and then re-execute the I/O instruction in the SMM handler by using the SMM I/O trap information. More likely, it can use the SMM I/O restart mechanism to cause the processor to restart the I/O instruction immediately after the RSM.

## 6.8.1 SMM I/O Trap

### SMM I/O Trap

Offset FEC0h

31																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												</
----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

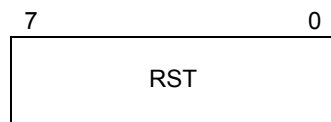
Bit	Name	Function	R/W
31–16	PORT	Trapped I/O Port	R
15–12	BRP	I/O Breakpoint Matches	R
11	TF	EFLAGS TF Value	R
10–7	reserved	RAZ	R
6	SZ32	Port Access was 32-bit	R
5	SZ16	Port Access was 16-bit	R
4	SZ8	Port Access was 8-bit	R
3	REP	Repeated Port Access	R
2	STR	String Based Port Access	R
1	V	I/O Trap Word Valid Bit	R
0	R/W	Port Access Type	R
		0 = Write (OUT instruction)	
		1 = Read (IN instruction)	

## 6.8.2 SMM I/O Restart Byte

The processor initializes the I/O trap restart slot to 00h on entry into SMM. If SMM is entered as a result of a trapped I/O instruction, the processor indicates the validity of the I/O instruction by setting or clearing bit 1 of the I/O trap doubleword at offset FEC0h in the SMM state save area. The SMM service routine should test bit 1 of the I/O trap doubleword to determine if a valid I/O instruction was being executed when entering SMM and before writing the I/O trap restart slot. If the I/O instruction is valid, the SMM service routine can safely rewrite the I/O trap restart slot with the value FFh, causing the processor to re-execute the trapped I/O instruction when the RSM instruction is executed. If the I/O instruction is invalid, writing the I/O trap restart slot has undefined results.

If a second SMI is asserted and a valid I/O instruction is trapped by the first SMM handler, the CPU services the second SMI prior to re-executing the trapped I/O instruction. The second entry into SMM will not have bit 1 of the I/O trap doubleword set, and the second SMM service routine must not rewrite the I/O trap restart slot.

During a simultaneous SMI I/O instruction trap and debug breakpoint trap, the processor first responds to the SMI and postpones recognizing the debug exception until after the RSM instruction returns from SMM. If the debug registers DR3–DR0 are used while in SMM, they must be saved and restored by the SMM handler. The processor automatically saves and restores DR7 and DR6. If the I/O trap restart slot in the SMM state save area contains the value FFh when the RSM instruction is executed, the debug trap does not occur until after the I/O instruction is re-executed.

**SMM I/O Restart Byte****Offset FEC8h**

Bit	Name	Function	R/W
7–0	RST	I/O Restart Byte 00h = Do not restart FFh = Restart I/O instruction	R/W

## 6.9 Exceptions and Interrupts in SMM

When SMM is entered, the processor masks INTR, NMI, SMI, INIT, and A20M interrupts. The processor clears the IF flag to disable INTR interrupts. To enable INTR interrupts within SMM, the SMM handler must set the IF flag to 1. A20M is disabled so that address bit 20 is always generated externally when in SMM.

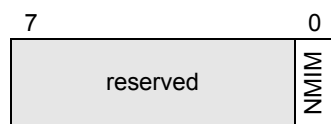
Generating an INTR interrupt is a method for unmasking NMI interrupts in SMM. The processor recognizes the assertion of NMI within SMM immediately after the completion of an IRET instruction. The NMI can be enabled by using an INTR interrupt. Once NMI is recognized within SMM, NMI recognition remains enabled until SMM is exited, at which point NMI masking is restored to the state it was in before entering SMM.

Because the IF flag is cleared when entering SMM, the HLT instruction should not be executed in SMM without first setting the IF bit to 1. Setting this bit to 1 enables the processor to exit the Halt state by means of an INTR interrupt.

The processor still responds to the DBREQ and STPCLK interrupts as well as to all exceptions that might be caused by the SMM handler.

### 6.10 NMI Mask

During entry into SMM, the NMI Mask byte at offset FECAh in the SMM state save area indicates whether NMI was masked when SMM was entered.

**NMI Mask****Offset FECAh**

Bit	Name	Function	R/W
7–1	reserved	SBZ	R/W
0	NMI	NMI Mask	R/W

## Field Description

**NMI Mask (NMIM)**—Bit 0. On entry:

0 = NMI not masked

1 = NMI Masked

## 6.11 Protected SMM and ASeg/TSeg

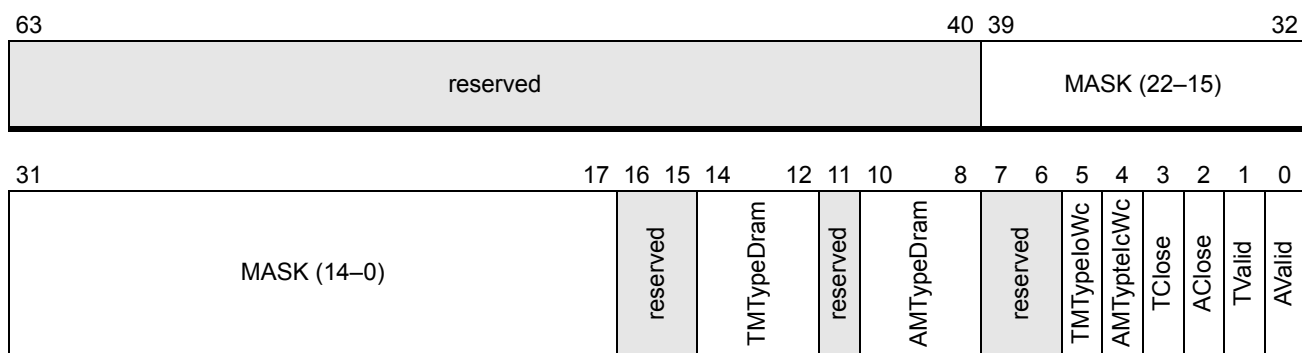
System management memory, SMRAM, is defined by two ranges. The ASeg range is located at a fixed range from A0000h–BFFFFh. The TSeg range is located at a variable base. The size of the TSeg range is controlled by a variable mask. SMRAM provides a safe location for system management code and data that is not readily accessible by applications. The SMM interrupt handler can be located in one of these two ranges if protection is needed, or it can be located outside these ranges in the rest of memory.

### 6.11.1 SMM\_MASK Register

This register holds the mask of the TSeg range as well as configuration information for both the fixed ASeg range and the variable TSeg range.

#### SMM\_MASK Register

MSR C001\_0113h



Bit	Name	Function	R/W
63–40	reserved	RAZ	
39–17	MASK	SMRAM TSeg Range Mask	R/W
16–15	reserved	RAZ	
14–12	TMTypeloDram	SMRAM TSeg Range Memory Type	R/W
11	reserved	RAZ	

Bit	Name	Function	R/W
10–8	AMTypeDram	SMRAM ASeg Range Memory Type	R/W
7–6	reserved	RAZ	
5	TMTypeloWc	Non-SMRAM TSeg Range Memory Type	R/W
4	AMTyptelcWc	Non-SMRAM ASeg Range Memory Type	R/W
3	TClose	Send TSeg Range Data Accesses to Non-SMRAM	R/W
2	AClose	Send ASeg Range Data Accesses to Non-SMRAM	R/W
1	TValid	Enable TSeg SMRAM Range	R/W
0	AValid	Enable ASeg SMRAM Range	R/W

**SMRAM TSeg Range Mask (MASK)**—Bits 39–17. Mask of the SMRAM TSeg Range.

**SMRAM TSeg Range Memory Type (TMTypeloWc)**—Bit 5. Memory Type for the SMRAM TSeg Range.

**SMRAM ASeg Range Memory Type (AMTypeDram)**—Bits 10–8. Memory Type for the SMRAM ASeg Range.

**Non-SMRAM TSeg Range Memory Type (TMTypeloWc)**—Bit 5. Memory Type for the non-SMRAM TSeg Range.

**Non-SMRAM ASeg Range Memory Type (AMTyptelcWc)**—Bit 4. Memory Type for the non-SMRAM ASeg Range.

**Send TSeg Range Data Accesses to Non-SMRAM (TClose)**—Bit 3. Send Data Accesses to the TSeg Range to Non-SMRAM.

**Send ASeg Range Data Accesses to Non-SMRAM (AClose)**—Bit 2. Send Data Accesses to the ASeg Range to Non-SMRAM.

**Enable TSeg SMRAM Range (TValid)**—Bit 1. Enables the TSeg SMRAM Range.

**Enable ASeg SMRAM Range (AValid)**—Bit 0. Enables the ASeg SMRAM Range.

## 6.11.2 SMM\_ADDR Register

This register holds the base of the variable TSeg range.

### SMM\_ADDR Register

**MSR C001\_0112h**

63	40	39	32
reserved			ADDR (22–15)
31	17	16	0
ADDR (14–0)		reserved	

Bit	Name	Function	R/W
63–40	reserved	RAZ	
39–17	ADDR	Base of the SMRAM TSeg Range	R/W
16–0	reserved	RAZ	

### 6.11.3 SMM ASeg

The SMM ASeg address range is A0000–BFFFFh. The ASeg is enabled by the AValid bit in the SMM\_MASK MSR.

When the ASeg is enabled, the MTRR memory maps are not used for the addresses in A0000–BFFFFh memory range. Instead, settings in the SMM\_MASK register and whether the processor is in SMM or not determine how those addresses are handled. If not in SMM, the addresses are mapped to I/O space and do not access DRAM. The memory type used is either UC or WC, based on the AMTypeIoWc bit in the SMM\_MASK register. When inside SMM, the accesses to the ASeg memory range are directed to DRAM.

The memory type is determined using the normal memory type encoding and is specified in AMTypeDram. Table 62 lists these types.

**Table 62. SMM ASeg-Enabled Memory Types**

		Out of SMM	
Address Range	In SMM	AMTypeIoWc=0	AMTypeIoWc=1
0–9FFFh	Normal MTRR	Normal MTRR	Normal MTRR
A0000–BFFFFh	DRAM, AMTypeDram	I/O, UC	I/O, WC
C0000h–Max	Normal MTRR	Normal MTRR	Normal MTRR

The SMM save state and code can be cached. After leaving SMM, the same addresses will bypass the caches and DRAM and access the I/O memory subsystem. This protects the SMM memory from corruption by programs outside of SMM.

### 6.11.4 SMM TSeg

The TSeg is similar to the ASeg except it provides more programability and therefore allows more space to be allocated in a different area of memory for the SMM handler. It is enabled by the TValid bit in the SMM\_MASK register. It uses the SMM\_ADDR register to specify its base. It can be used with ASeg or by itself. When it is used with ASeg, the two spaces should not overlap. If they do overlap, the memory types should be consistent in both the ASeg map and the TSeg map for the overlapping addresses. Otherwise, undefined behavior will occur.

The SMM TSeg begins at the address specified in SMM\_ADDR concatenated with 0s in the lower 17 bits, forcing the TSeg to begin on a 128-Kbyte boundary. The ending of TSeg is specified by the SMM\_Mask[39:17]. The SMM\_Mask and SMM\_ADDR function as the variable-range MTRRs.

Each address generated by the processor is ANDed with the SMM\_MASK and compared to the SMM\_ADDR ANDed with the mask. If the values are equal, the address is within the TSeg range. For example, suppose you wish to create a TSeg starting at the 1-Mbyte address boundary and extending 256 Kbytes. The SMM\_ADDR register would be set to 0010\_0000h and the SMM\_MASK to FFFC\_0002h. This will make the TSeg range from 0010\_0000–0013\_FFFh.

The TSeg memory type table is similar to the ASeg memory type table. When not in SMM and TSeg is enabled, the addresses within the TSeg range are directed to I/O space with either a UC memory type or a WC memory type, based on the TMTypeloWc bit in the SMM\_MASK register. When within SMM, the accesses are directed to DRAM with a memory type specified by TMTypEDram.

**Table 63. SMM TSeg-Enabled Memory Types**

Address Range	In SMM	Out of SMM	
		TMTypeloWc=0	TMTypeloWc=1
<TSeg Range	Normal MTRR	Normal MTRR	Normal MTRR
TSeg Range	DRAM, TMTypEDram	I/O, UC	I/O, WC
> TSeg Range	Normal MTRR	Normal MTRR	Normal MTRR

### 6.11.5 Closing SMM

Sometimes within SMM code with ASeg or TSeg enabled, there is a requirement to access the I/O space at the same address as the current SMM segment. That is typically only accessible outside of SMM. To accomplish this function, the Aclose and Tclose bits from SMM\_MASK register are used. When the Aclose bit is set, data cache accesses to the ASeg that would normally go to DRAM are redirected to I/O, with the memory type specified by AMTypeloWc.

The same function applies to the TSeg. Instruction cache accesses and Page Directory/Table accesses still access the SMM code in DRAM. When the SMM handler is done accessing the I/O space, it must clear the appropriate close bit. Failure to do so and then issuing an RSM will probably cause the processor to enter shutdown, as the save state will be read from I/O space.

### 6.11.6 Locking SMM

The SMM registers can be locked by setting the SMMLOCK (HWCR, bit 0). Once set, the SMM\_BASE, the SMM\_ADDR, all but the two close bits of SMM\_MASK and the RSMSPCYCDIS, SMISPCYCDIS, and SMMLOCK bits of HWCR are locked and cannot be changed. The only way to unlock the SMM registers is to assert reset. This provides security to the SMM mechanism. The BIOS can lock the SMM environment after setting it up so that it can not be tampered with.



## 6.12 SMM Special Cycles

Special cycles can be initiated on entry and exit from SMM to acknowledge to the system that these transitions are occurring. This allows secure spaces external to the processor to be secured within the SMM environment as well. It also helps with synchronizing the SMI interrupt across multiple processor systems. These special cycles can be disabled by setting the SMISPCYCDIS to 1 to disable the entry special cycle and setting the RSMSPCYCDIS bit to 1 to disable the exit special cycle.



## 7 HyperTransport™ Technology Configuration and Enumeration

---

An AMD Athlon™ 64 and AMD Opteron™ Processor is a node connected to other nodes with coherent HyperTransport™ links. Function 0 HyperTransport technology configuration register values for each node must be initialized. A node is identified in the Node ID register (Function 0, Offset 60h) with a number from 0 to 7. Routing tables are used by the node to decide whether to accept a packet and/or forward it through any or all of its HyperTransport links.

Three examples of coherent HyperTransport initialization sequence are shown in this chapter: 1-node initialization, 2-node initialization, and generic initialization that can enumerate any number of nodes. This chapter also provides information on how to determine if a given routing table is valid.

### 7.1 Initial Configuration Steps

For coherent HyperTransport technology initialization, the following steps are required before enumeration:

1. Coherent HyperTransport initialization is only performed by Node 0 (BSP). All the other nodes (APs) should bypass the HyperTransport initialization process.
2. Clearing RoutTblDis (Function 0, Offset 6Ch) enables the routing table for Node 0 and allows access to the memory controller on Node 0.
3. Node 0 is by definition connected to the HyperTransport I/O Hub, which means that Node 0 owns the compatibility chain. The link number that connects to the HyperTransport I/O Hub should be written to SbLink (Function 0, Offset 64h).
4. Count coherent HyperTransport links on Node 0.

To perform step 4, it is necessary to detect whether each HyperTransport link on a node is connected and if the connected link type is coherent or noncoherent. The AMD Opteron™ processor supports up to three links, and the AMD Athlon™ 64 processor supports one link. For each HyperTransport link, the following steps are performed to detect the link connection status and type:

1. Check whether LinkConPend (Function 0, Offset 98h, D8h, B8h) is set.
2. If the LinkConPend is clear, check whether LinkCon (Function 0, Offset 98h, D8h, B8h) is set. If the Link Connected bit is set, the testing port is connected to other node.
3. Check whether InitComplete (Function 0, Offset 98h, D8h, B8h) is set.
4. If InitComplete is set, check NC (Function 0, Offset 98h, D8h, B8h) bit. The bit value 0 indicates a coherent HyperTransport link, while value 1 indicates a noncoherent link.
5. Record the number of coherent and noncoherent ports and their respective port numbers.

## 7.2 One-Node Coherent HyperTransport™ Technology Initialization

The number of nodes that exist in the system is determined on page 235. The following configuration steps should be executed in single (one-node) processor systems:

1. Set LimitCldtCfg (Function 0, Offset 68h) to enable limiting the extent of coherent HyperTransport configuration space based on the number of nodes in the system.
2. Disable read/write/fill probes in Function 0, Offset 68h for single-core one-node systems, since single processor systems do not need probes. For dual-core one-node systems, enable read/write/fill probes in Function 0, Offset 68h.

## 7.3 Two-Node Coherent HyperTransport™ Technology Initialization

For two-node systems, Node IDs and Routing Tables need to be written for both nodes, as follows.

1. Initialize Node 0 Routing Table rows 0 and 1. Based on the coherent HyperTransport link number on Node 0.
2. Verify if Node 0-to-Node 1 link is established by reading Vendor ID/Device ID of Node 1.
3. Initialize Node 1 Routing Table row 0 and 1 using the coherent HyperTransport link number on Node 1.
4. Initialize the node ID of Node 1 by programming NodeId (Function 0, Offset 60h).
5. Write CPU Count and Node Count to CpuCnt and NodeCnt (Function 0, Offset 60h).
6. Set LimitCldtCfg (Function 0, Offset 68h) for Node 0 and Node 1.
7. Clear RouteTblDis (Function 0, Offset 6Ch) for Node 1.

## 7.4 Generic HyperTransport™ Technology Configuration

For multi-node systems, Node ID and Routing Tables need to be written for all nodes, as follows. This algorithm assumes the BIOS knows the topology of the nodes and how each link between nodes is routed on the system board.

1. Initialize Node 0 Routing Table rows 0 through N based on the coherent HyperTransport link numbers on Node 0.
2. For each node n in the system (starting with the nodes connected to Node 0 and working outward to the nodes furthest from Node 0) repeat steps a through e.
  - a. Verify if Node 0-to-Node n link is established by reading Vendor ID/Device ID of Node n.

- If the link node n is not established, disable the Routing Table on each node and go back to step 1 and build the Routing Tables for each node based on the new configuration.
- b. Poll the RequestDisable bit (Function 0, Offset 60h) on Node n until set.
- c. Initialize Node n Routing Table row 0 through N. Using the coherent HyperTransport link numbers on Node n.
- d. Initialize the node ID of Node n by programming NodeId (Function 0, Offset 60h).
- e. Clear RouteTblDis (Function 0, Offset 6Ch) for Node 1.
- 3. Write CPU Count and Node Count to CpuCnt and NodeCnt (Function 0, Offset 60h) of each Node.
- 4. Set LimitCldtCfg (Function 0, Offset 68h) bit for each Node.

## 7.5 Rules for Valid Routing Tables

This section provides rules for determining if a given routing table is valid. A routing table is valid if and only if the routing table is deadlock-free and probes are delivered only once.

**I** A routing table is deadlock-free if it contains no Open Paths and no two-hop cycles.

An Open Path is a routing path between nodes that passes through one or more nodes that contains a subpath that is not a routing path in the routing table. For example if the routing path between Nodes 0 and 2 in Figure 5 was Node 0->Node 1->Node 3->Node 2 and the routing path between Nodes 3 and 2 was not Node 3->Node 2 then the routing path between Nodes 0 and 2 would be open because the subpath Node 3->Node 2 is not a path in the routing table.

A two-hop cycle is a group of two hop routing paths (routing paths between two nodes that pass through a third node) such that the first and second nodes in the each two hop routing path are also the second and third nodes in a two hop routing path in the group.

### 7.5.1 Two-Hop Cycle Example

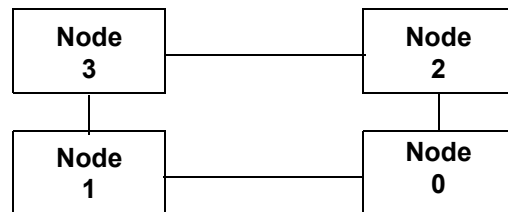
Consider the four node configuration shown in Figure 5. A two-hop cycle occurs in the configuration if the routing table is configured with the following routing paths:

- The routing path from Node 0 to Node 3 is: Node 0->Node 1->Node 3.
- The routing path from Node 1 to Node 2 is: Node 1->Node 3->Node 2.
- The routing path from Node 2 to Node 1 is: Node 2->Node 0->Node 1.
- The routing path from Node 3 to Node 0 is: Node 3->Node 2->Node 0.

To break this cycle, at least one (but no more than three) of these routing paths must be modified to use a different intermediate node. Reconfiguring the routing paths as follows eliminates the two-hop cycle.

- The routing path from Node 0 to Node 3 is: Node 0->Node 1->Node 3.

- The routing path from Node 1 to Node 2 is: Node 1->Node 3 to Node 2.
- The routing path from Node 2 to Node 1 is: Node 2->Node 0->Node 1.
- The routing path from Node 3 to Node 0 is: Node 3->Node 1->Node 0.



**Figure 5. A Four-Node Configuration**

## 8 Advanced Programmable Interrupt Controller (APIC)

---

All AMD Athlon™ 64 and AMD Opteron™ Processor-based systems must support 64-bit operating systems. As a result, all platforms must support APIC and declare it to the operating system in the appropriate tables.

The local APIC contains logic to receive interrupts from a variety of sources and to send interrupts to other local APICs, as well as registers to control its behavior and report status.

Interrupts can be received from:

- HyperTransport™ I/O devices, including the I/O hub (I/O APICs)
- Other local APICs (inter-processor interrupts)
- APIC timer
- Performance counters
- Legacy local interrupts from the I/O hub (INTR and NMI)
- APIC internal errors

The APIC timer, performance counters, local interrupts, and internal errors are all considered local interrupt sources, and their routing is controlled by local vector table entries. These entries assign a message type and vector to each interrupt, allow them to be masked, and track the status of the interrupt.

I/O and inter-processor interrupts have their message type and vector assigned at the source and are unaltered by the local APIC. They carry a destination field and a mode bit that together determine which local APICs will accept them. The destination mode (DM) bit specifies if the interrupt request packet should be handled in Physical or Logical destination mode.

- In physical destination mode, if the destination field is FFh, the interrupt is a broadcast and is accepted by all local APICs. Otherwise, the interrupt is only accepted by the local APIC whose APIC ID matches the destination field of the interrupt. Physical mode allows up to 255 APICs to be addressed individually.

- In logical destination mode, a local APIC accepts interrupts selected by the logical destination register (LOG\_DEST) and the destination field of the interrupt using either cluster or flat format, as configured by the destination format register.
  - If flat destinations are in use, bits 7–0 of the LOG\_DEST field are checked against bits 7–0 of the arriving interrupt's Destination Field. If any bit position is set in both fields, this APIC is a valid destination. Flat format allows up to 8 APICs to be addressed individually.
  - If cluster destinations are in use, bits 7–4 of the LOG\_DEST field are checked against bits 7–4 of the arriving interrupt's destination field, identifying the cluster. If all of bits 7–4 match, then bits 3–0 of the LOG\_DEST and the interrupt destination are checked for any bit positions that are set in both fields, choosing processors within the cluster. If both conditions are met, this APIC is a valid destination. Cluster format allows 15 clusters of 4 APICs each to be addressed.
  - In both flat and cluster formats, if the destination field is FFh, the interrupt is a broadcast and is accepted by all local APICs.

## 8.1 Interrupt Delivery

SMI, NMI, INIT, Startup, and External interrupts are classified as non-vectorized interrupts.

When an APIC accepts a non-vectorized interrupt, it is handled directly by the processor instead of being queued in the APIC. When an APIC accepts a fixed or lowest priority interrupt, it sets the bit in the interrupt request register (IRR) corresponding to the vector in the interrupt. (For local interrupt sources, this comes from the vector field in that interrupt's local vector table entry.) If a subsequent interrupt with the same vector arrives when the corresponding IRR bit is already set, the two interrupts are collapsed into one. Vectors 0–15 are reserved.

## 8.2 Vectored Interrupt Handling

The task priority register (TPR\_PRI) and processor priority register (PROC\_PRI) each contain an 8-bit priority, divided into a main priority (bits 7–4) and a priority sub-class (3–0). The task priority is assigned by software to set a threshold priority at which the processor will be interrupted.

To calculate processor priority, an 8-bit ISR vector is set based on the highest bit set in the in-service register (ISR). Like the TPR\_PRI and PROC\_PRI, this vector is divided into a main priority (7–4) and priority sub-class (3–0). The main priority of the TPR\_PRI and ISR are compared and the highest is the PROC\_PRI, as follows:

```
If (TPR_PRI[7:4] >= ISRVect[7:4]) PROC_PRI = TPR_PRI
Else                                     PROC_PRI = {ISRVect[7:4], 0h}
```

The resulting processor priority is used to determine if any accepted interrupts (indicated by IRR bits) are high enough priority to be serviced by the processor. When the processor is ready to service an interrupt, the highest bit in the IRR is cleared, and the corresponding bit is set in the ISR. The



corresponding trigger mode register (TMR) bit is set if the interrupt is level-triggered and cleared if edge-triggered.

When the processor has completed service for an interrupt, it performs a write to the end of interrupt (EOI) register, clearing the highest ISR bit and causing the next-highest interrupt to be serviced. If the corresponding TMR bit is set, EOI messages will be sent to all APICs to complete service of the interrupt at the source.

## 8.3 Spurious Interrupts

In the event that the task priority is set to or above the level of the interrupt to be serviced, the local APIC will deliver a spurious interrupt vector to the processor, as specified by the spurious interrupt vector register. The ISR will be unchanged and no EOI will occur.

### 8.3.1 Spurious Interrupts Caused by Timer Tick Interrupt

A typical interrupt is asserted until it is serviced. Interrupt is deasserted when software clears the interrupt status bit within the interrupt service routine. Timer tick interrupt is an exception, since it is deasserted regardless of whether it is serviced or not. A BIOS programs the 8254 Programmable Timer to generate an 18.2 Hz square wave. This square wave represents the timer tick interrupt (PITIRQ) and in some interrupt configurations it is connected to the 8259 Programmable Interrupt Controller (PIC) IRQ0 input.

The processor is not always able to service interrupts immediately (i.e. when interrupts are masked by clearing EFLAGS.IM or when the processor is in debug mode). The method of interrupt delivery to the processor (wire or messages) determines system behavior for the timer tick interrupt connected to the PIC after the processor has not been able to service it for some time. The PIC output INTR is identical to PITIRQ when interrupts are not serviced.

If interrupts are delivered to the processor using a wire, and the processor is not able to service the timer tick interrupt for some time, timer tick interrupts asserted during that time will be lost. The following cases are possible when the processor is ready to service interrupts:

- INTR is deasserted and it is not serviced by the processor. This will happen almost 50 percent of the time.
- INTR is asserted and it is serviced by the processor. This will happen almost 50 percent of the time.
- INTR is asserted, and it is detected by the processor. The processor sends the interrupt acknowledge cycle, but when the PIC receives it, INTR is deasserted, and the PIC sends a spurious interrupt vector. This will happen a very small percentage of the time.

The probability of spurious interrupts, when interrupts are delivered to the processor using a wire, is very low. An example of a configuration that uses wire to deliver interrupts to the processor has PITIRQ connected to PIC IRQ0, and PIC INTR connected to LINT0 of an AMD Athlon processor.

If interrupts are delivered to the processor using messages, and the processor is not able to service the timer tick interrupt for an extended period of time, the INTR caused by the first timer tick interrupt asserted during that time is delivered to the local APIC in ExtInt mode and latched, and the subsequent timer tick interrupts are lost. The following cases are possible when the processor is ready to service interrupts:

- An ExtInt interrupt is pending, and INTR is asserted. This results in timer tick interrupt servicing. This occurs 50 percent of the time.
- An ExtInt interrupt is pending, and INTR is deasserted. The processor sends the interrupt acknowledge cycle, but when the PIC receives it, INTR is deasserted, and the PIC sends a spurious interrupt vector. This occurs 50 percent of the time.

There is a 50 percent probability of spurious interrupts when interrupts are delivered to the processor using messages. An example of an AMD Athlon™ 64 or AMD Opteron™ processor configuration that uses messages to deliver interrupts to the processor has PITIRQ connected to PIC IRQ0, PIC INTR connected to IOAPIC IRQ0, and interrupts are delivered to the processor by HyperTransport messages. HyperTransport messages are always used to deliver PIC interrupts to the processor in a system with AMD Athlon™ 64 or AMD Opteron™ processors. An example of an AMD Athlon processor configuration that uses messages to deliver interrupts to the processor has PITIRQ connected to PIC IRQ0, PIC INTR connected to IOAPIC IRQ0, and interrupts are delivered to the processor by APIC 3-wire messages.

## 8.4 Lowest-Priority Arbitration

Fixed and non-vectored interrupts are accepted by their destination APICs without arbitration.

Delivery of lowest-priority interrupts requires all APICs to arbitrate to determine which one will accept the interrupt. If focus processor checking is enabled (bit 9 of the spurious interrupt vector register cleared), then the focus processor for an interrupt will always accept the interrupt. A processor is the focus of an interrupt if it is already servicing that interrupt (corresponding ISR bit is set) or if it already has a pending request for that interrupt (corresponding IRR bit is set). If there is no focus processor for an interrupt, or focus processor checking is disabled, then each APIC will calculate an arbitration priority value, and the one with the lowest result will accept the interrupt.

To calculate arbitration priority, an 8-bit IRR Vector is set based on the highest bit set in the IRR. Like the ISR, TPR\_PRI, and PROC\_PRI, this vector is divided into a main priority (7–4) and priority subclass (3–0). The main priority of the TPR\_PRI, ISR, and IRR are compared and the highest is the new ARB\_PRI, as follows:

```
If    (TPR_PRI[7:4] >= IRRVect[7:4] and
      TPR_PRI[7:4] >  ISRVect[7:4])   ARB_PRI = TPR_PRI
Else if (IRRVect[7:4] > ISRVect[7:4]) ARB_PRI = {IRRVect[7:4], 0h}
Else                                     ARB_PRI = {ISRVect[7:4], 0h}
```

## 8.5 Inter-Processor Interrupts

In order to redirect an interrupt to another processor, originate an interrupt to another processor, or allow a processor to interrupt itself, the interrupt command register (ICR) provides a mechanism for generating interrupts. A write to the low doubleword of the ICR causes an interrupt to be generated, with the properties specified by the ICR low and ICR high fields.

## 8.6 APIC Timer Operation

The local APIC contains a 32-bit timer, controlled by the Timer LVT entry, initial count, and divide configuration registers. The processor bus clock is divided by the value in the timer divide configuration register to obtain a time base for the timer. When the timer initial count register is written, the value is copied into the timer current count register. The current count register is decremented at the rate of the divided clock. When the count reaches 0, a timer interrupt is generated with the vector specified in the Timer LVT entry. If the Timer LVT entry specifies periodic operation, the current count register is reloaded with the initial count value, and it continues to decrement at the rate of the divided clock. If the mask bit in the timer LVT entry is set, timer interrupts are not generated.

## 8.7 State at Reset

At power-up or reset, all registers take on the values listed in their descriptions. SMI, NMI, INIT, Startup, and LINT interrupts may be accepted.

When the APIC is software-disabled, pending interrupts in the ISR and IRR are held, but further fixed, lowest-priority, and ExtInt interrupts will not be accepted. All LVT entry mask bits are set and cannot be cleared.

When a processor accepts an INIT interrupt, the APIC is reset as at power-up, with the exception that APIC\_ID is unaffected.

## 8.8 Register Summary

All APIC registers are accessed with memory reads and writes of (APIC\_BASE+Offset). APIC\_BASE is MSR 001Bh and defaults to 00\_FEE0\_0000h.

The APIC registers and their offsets are listed in Table 64.

**Table 64. APIC Register Summary**

Offset (Hex)	Mnemonic	Name
20	APIC_ID	APIC ID Register
30	APIC_VER	APIC Version Register
80	TPR_PRI	Task Priority Register
90	ARB_PRI	Arbitration Priority Register
A0	PROC_PRI	Processor Priority Register
B0	EOI	End Of Interrupt Register
D0	LOG_DEST	Logical Destination Register
E0	DEST_FORMAT	Destination Format Register
F0	SPUR_INTR_VEC	Spurious Interrupt Vector Register
100-170	ISR	In-Service Registers
180-1F0	TMR	Trigger Mode Registers
200-270	IRR	Interrupt Request Registers
280	ERR_STAT	Error Status Register
300	ICRLO	Interrupt Command Register Low (bits 31–0)
310	ICRHI	Interrupt Command Register High (bits 63–32)
320	TIMER_LVT	Timer Local Vector Table Entry
340	PERF_CNT_LVT	Performance Counter Local Vector Table Entry
350	LINT0_LVT	Local Interrupt 0 Local Vector Table Entry
360	LINT1_LVT	Local Interrupt 1 Local Vector Table Entry
370	ERROR_LVT	Error Local Vector Table Entry
380	INIT_CNT	Timer Initial Count Register
390	CURR_CNT	Timer Current Count Register
3E0	TIMER_DVD_CFG	Timer Divide Configuration Register
400	EXT_APIC_FEAT	Extended APIC Feature Register

**Table 64. APIC Register Summary (Continued)**

410	EXT_APIC_CTRL	Extended APIC Control Register
-----	---------------	--------------------------------

### 8.8.1 APIC ID Register

#### APIC\_ID Register

Offset 20h

31	24	23	0
APICID	reserved		

Bit	Name	Function	R/W	Reset
31–24	APICID	APIC Identification	R/W	Node ID
23–0	reserved		R/O	00_0000h

#### Field Descriptions

**APIC Identification (APICID)**—Bits 31–24. Software must ensure that all APICs are assigned unique APIC IDs. When both ApicExtId and ApicExtBrdCst in the HyperTransport™ Transaction Control Register are set, all 8 bits of APIC ID are used. When either ApicExtID or ApicExtBrdCst is clear, only bits 3–0 of APIC ID are used, and bits 7–4 are reserved. Node ID is initially 00h if this is the boot strap processor or 07h for all other nodes.

### 8.8.2 APIC Version Register

#### APIC\_VER Register

Offset 30h

31	24	23	16	15	8	7	0
reserved	MaxLVTEEntry		reserved	Version			

Bit	Name	Function	R/W	Reset
31–24	reserved		R/O	00h
23–16	MaxLVTEEntry	Maximum LVT Entry	R/O	04h
15–8	reserved		R/O	00h
7–0	Version	Version	R/O	10h

#### Field Descriptions

**Max LVT Entry**—Bits 23–16. This field indicates the number of entries in the Local Vector Table minus one.

**Version**—Bits 7–0. This field indicates the version number of this APIC implementation.

### 8.8.3 Task Priority Register

#### TPR\_PRI Register

Offset 80h



Bit	Name	Function	R/W	Reset
31–8	reserved		R/O	0000_00h
7–0	Priority	Priority	R/W	00h

#### Field Descriptions

**Priority**—Bits 7–0. This field is assigned by software to set a threshold priority at which the processor will be interrupted.

### 8.8.4 Arbitration Priority Register

#### ARB\_PRI Register

Offset 90h



Bit	Name	Function	R/W	Reset
31–8	reserved		R/O	0000_00h
7–0	Priority	Priority	R/O	00h

#### Field Descriptions

**Priority**—Bits 7–0. This field indicates the processor's current priority, for a task being serviced, an interrupt being serviced, or an interrupt that is pending, and is used to arbitrate between processors to determine which will accept a lowest-priority interrupt request.

## 8.8.5 Processor Priority Register

### PROC\_PRI Register

Offset A0h



Bit	Name	Function	R/W	Reset
31–8	reserved		R/O	0000_00h
7–0	Priority	Priority	R/O	00h

### Field Descriptions

**Priority**—Bits 7–0. This field indicates the processor's current priority servicing a task or interrupt, and is used to determine if any pending interrupts should be serviced.

## 8.8.6 End of Interrupt Register

This register is written by the software interrupt handler to indicate that servicing of the current interrupt is complete.

### EOI Register

Offset B0h



Bit	Name	Function	R/W	Reset
31–0	reserved		W/O	XXXX_XXXXh

## 8.8.7 Logical Destination Register

### LOG\_DEST Register

Offset D0h



Bit	Name	Function	R/W	Reset
31–24	Destination	Destination	R/W	00h
23–0	reserved		R/O	00_0000h

## Field Descriptions

**Destination (Destination)**—Bits 31–24. This field contains this APIC's destination identification, and is used to determine which interrupts should be accepted.

## 8.8.8 Destination Format Register

### DEST\_FORMAT Register

Offset E0h



Bit	Name	Function	R/W	Reset
31–28	Format	Format	R/W	Fh
27–0	reserved		R/O	FFF_FFFh

## Field Descriptions

**Format (Format)**—Bits 31–28. 0h and Fh are the only allowed values. This field controls which format to use when accepting interrupts with a logical destination mode.

0h = Cluster destinations are used.

Fh = Flat destinations are used.

## 8.8.9 Spurious Interrupt Vector Register

### SPUR\_INTR\_VEC Register

Offset F0h



Bit	Name	Function	R/W	Reset
31–10	reserved		R/O	0000_00h
9	FocusDisable	Focus Disable	R/W	0



Bit	Name	Function	R/W	Reset
8	APICSWEn	APIC Software Enable	R/W	0
7–0	Vector	Vector	R/W	FFh

## Field Descriptions

**Focus Disable (FocusDisable)**—Bit 9. This bit disables focus processor checking during lowest-priority arbitrated interrupts.

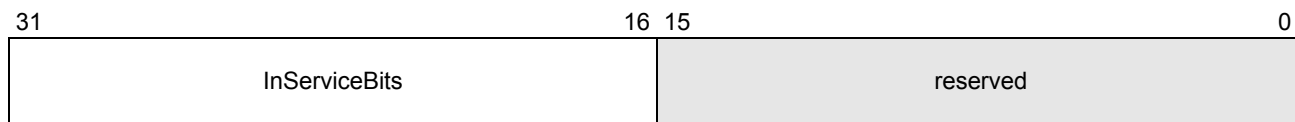
**APIC Software Enable (APICSWEn)**—Bit 8. When APICSWEn is cleared, SMI, NMI, INIT, Startup, and LINT interrupts may be accepted, pending interrupts in the ISR and IRR are held, but further Fixed, Lowest-Priority, and ExtInt interrupts will not be accepted. All LVT entry Mask bits are set and cannot be cleared.

**Vector**—Bits 7–0. When ApicExtSpur in the HyperTransport™ Transaction Control Register is set, bits 3–0 of Vector are writable. When ApicExtSpur is clear, bits 3–0 are read-only 1111b. This field contains the vector that is sent to the processor in the event of a spurious interrupt.

## 8.8.10 In-Service Registers

### ISR Register

Offset 100h



Bit	Name	Function	R/W	Reset
31–16	InServiceBits	In-Service Bits	R/O	0000h
15–0	reserved		R/O	0000h

## Field Descriptions

**In-Service Bits (InServiceBits)**—Bits 31–16. These bits are set when the corresponding interrupt is being serviced by the CPU.

### ISR Registers

Offsets 170h, 160h, 150h, 140h, 130h, 120h, 110h



Bit	Name	Function	R/W	Reset
31–0	InServiceBits	In-Service Bits	R/O	0000_0000h

## Field Descriptions

**In-Service Bits (InServiceBits)**—Bits 31–0. These bits are set when the corresponding interrupt is being serviced by the CPU. Interrupts are mapped as follows:

ISR	Interrupt number
Offset 110h	63–32
Offset 120h	95–64
Offset 130h	127–96
Offset 140h	159–128
Offset 150h	191–160
Offset 160h	223–192
Offset 170h	255–224

### 8.8.11 Trigger Mode Registers

#### TMR Register

Offset 180h

31	16	15	0
TriggerModeBits			reserved

Bit	Name	Function	R/W	Reset
31–16	TriggerModeBits	Trigger Mode Bits	R/O	0000h
15–0	reserved		R/O	0000h

## Field Descriptions

**Trigger Mode Bits (TriggerModeBits)**—Bits 31–16. The Trigger Mode bit for each corresponding interrupt is updated when an interrupt enters servicing. It is 0 for edge-triggered interrupts and 1 for level-triggered interrupts.

#### TMR Registers

Offsets 1F0h, 1E0h, 1D0h, 1C0h, 1B0h, 1A0h, 190h

31	0
Trigger Mode Bits	

Bit	Name	Function	R/W	Reset
31–0	TriggerModeBits	Trigger Mode Bits	R/O	0000_0000h

## Field Descriptions

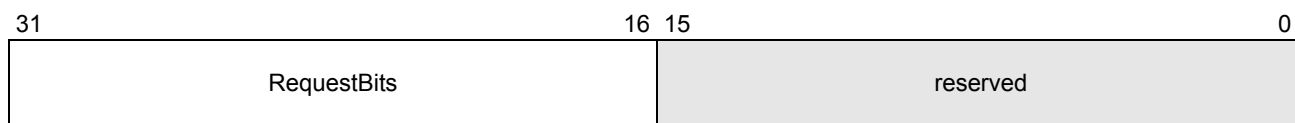
**Trigger Mode Bits (TriggerModeBits)**—Bits 31–0. The Trigger Mode bit for each corresponding interrupt is updated when an interrupt enters servicing. It is 0 for edge-triggered interrupts and 1 for level-triggered interrupts. Interrupts are mapped as follows:

TMR	Interrupt number
Offset 190h	63–32
Offset 1A0h	95–64
Offset 1B0h	127–96
Offset 1C0h	159–128
Offset 1D0h	191–160
Offset 1E0h	223–192
Offset 1F0h	255–224

## 8.8.12 Interrupt Request Registers

### IRR Register

Offset 200h



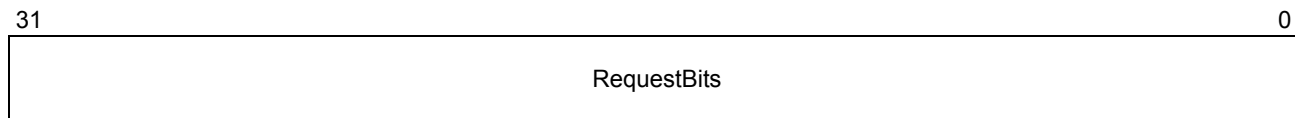
Bit	Name	Function	R/W	Reset
31–16	RequestBits	Interrupt Requests Bits	R/O	0000h
15–0	reserved		R/O	0000h

## Field Descriptions

**Interrupt Request Bits (RequestBits)**—Bits 31–16. Request bits are set when the corresponding interrupt is accepted by the APIC.

### IRR Registers

Offsets 270h, 260h, 250h, 240h, 230h, 220h, 210h



Bit	Name	Function	R/W	Reset
31–0	RequestBits	Interrupt Requests Bits	R/O	0000_0000h

## Field Descriptions

**Interrupt Request Bits (RequestBits)**—Bits 31–0. Request bits are set when the corresponding interrupt is accepted by the APIC. Interrupts are mapped as follows:

IRR	Interrupt number
Offset 210h	63–32
Offset 220h	95–64
Offset 230h	127–96
Offset 240h	159–128
Offset 250h	191–160
Offset 260h	223–192
Offset 270h	255–224

### 8.8.13 Error Status Register

This register must be written to trigger an update before it can be read. Each write causes the internal error state to be loaded into this register, clearing the internal error state. A second write before another error occurs causes this register to be cleared.

## ERR\_STAT Register

## Offset 280h

31	8	7	6	5	4	3	2	1	0					
reserved								IllegalRegAddr	RcvdIllegalVector	SentIllegalVector	reserved	RcvAcceptError	SentAcceptError	reserved

Bit	Name	Function	R/W	Reset
31–8	reserved		R/O	0000_00h
7	IllegalRegAddr	Illegal Register Address	W/R	0
6	RcvdIllegalVector	Received Illegal Vector	W/R	0
5	SentIllegalVector	Sent Illegal Vector	W/R	0
4	reserved		R/O	0
3	RcvAcceptError	Receive Accept Error	W/R	0
2	SendAcceptError	Send Accept Error	W/R	0
1–0	reserved		R/O	00b

## Field Descriptions

**Illegal Register Address (IllegalRegAddr)**—Bit 7. This bit indicates that an access to a nonexistent register location within this APIC was attempted.

**Received Illegal Vector (RcvdIllegalVector)**—Bit 6. This bit indicates that this APIC received a message with an illegal Vector (00h to 0Fh for fixed and lowest priority interrupts).

**Sent Illegal Vector (SentIllegalVector)**—Bit 5. This bit indicates that this APIC attempted to send a message with an illegal Vector (00h to 0Fh for fixed and lowest priority interrupts).

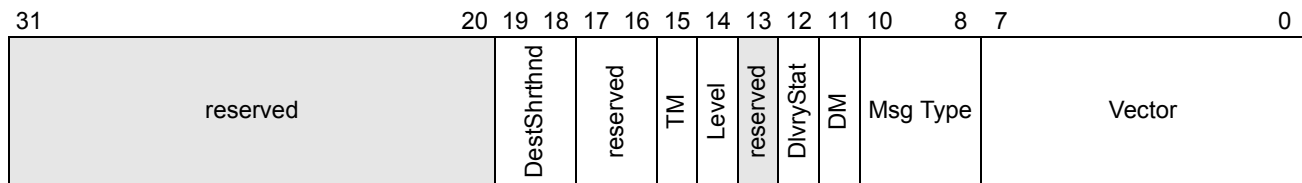
**Receive Accept Error (RcvAcceptError)**—Bit 3. This bit indicates that a message received by this APIC was not accepted by this or any other APIC.

**Send Accept Error (SendAcceptError)**—Bit 2. This bit indicates that a message sent by this APIC was not accepted by any APIC.

## 8.8.14 Interrupt Command Register Low

### ICRLO Register

Offset 300h



Bit	Name	Function	R/W	Reset
31–20	reserved		R/O	000h
19–18	DestShrthnd	Destination Shorthand	R/W	00b
17–16	reserved		R/O	00b
15	TM	Trigger Mode	R/W	0
14	Level	Level	R/W	0
13	reserved		R/O	0
12	DivryStat	Delivery Status	R/O	0
11	DM	Destination Mode	R/W	0
10–8	MsgType	Message Type	R/W	000b
7–0	Vector	Vector	R/W	00h

### Field Descriptions

**Destination Shorthand (DestShrthnd)**—Bits 19–18. This field provides a quick way to specify a destination for a message. If All Including Self or All Excluding Self are used, then DM is ignored and physical is automatically used.

00b =Destination Field

01b =Self

10b =All Including Self

11b =All Excluding Self (Note that this sends a message with a destination encoding of all 1's, so if lowest priority is used, the message could end up being reflected back to this APIC.)

**Trigger Mode (TM)**—Bit 15. This bit can be 0 for Edge or 1 for Level.

**Level (Level)**—Bit 14. This bit can be 0 for deasserted or 1 or asserted.

**Delivery Status (DlvryStat)**—Bit 12. This bit is set to indicate that the interrupt has not yet been accepted by the destination CPU(s).

**Destination Mode (DM)**—Bit 11. This bit can be 0 for Physical or 1 for Logical.  
**Message Type (MsgType)**—Bits 10–8. The encoding for this field is as follows:

- 000b = Fixed
- 001b = Lowest Priority
- 010b = SMI
- 011b = Remote Read
- 100b = NMI
- 101b = INIT
- 110b = Startup
- 111b = External Interrupt

**Note:** Not all combinations of ICR fields are valid. Only those combinations listed in Table 65 are valid.

**Table 65. Valid Combinations of ICR Fields**

Message Type (MsgType)	Trigger Mode (TM)	Level (Lvl)	Destination Shorthand (DestShrthnd)
Fixed	Edge	X	X
	Level	Assert	X
Lowest Priority, SMI, NMI, INIT	Edge	X	Dest or All Excluding Self
	Level	Assert	Dest or All Excluding Self
Startup	X	X	Dest or All Excluding Self
<b>Note:</b> X indicates a “don’t care”.			

**Vector (Vector)**—Bits 7–0. This field contains the vector that will be sent for this interrupt source.

## 8.8.15 Interrupt Command Register High

**ICRHI Register**

**Offset 310h**

63	56	55	32
DestinationField	reserved		

Bit	Name	Function	R/W	Reset
63–56	DestinationField	Destination Field	R/W	00h
55–32	reserved		R/O	00_0000h

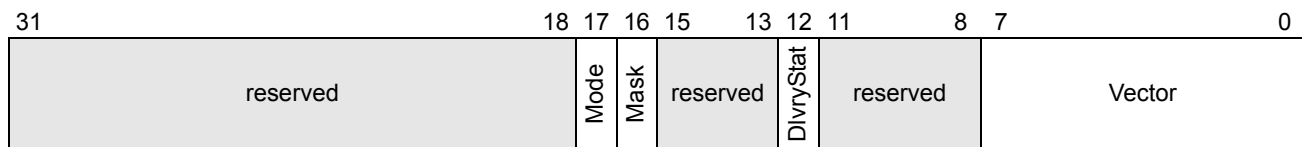
## Field Descriptions

**Destination Field (DestinationField)**—Bits 63–56. This field contains the destination encoding used when the destination shorthand is 00b.

## 8.8.16 Timer Local Vector Table Entry

### TIMER\_LVT Register

Offset 320h



Bit	Name	Function	R/W	Reset
31–18	reserved		R/O	0000h
17	Mode	Mode	R/W	0
16	Mask	Mask	R/W	1
15–13	reserved		R/O	000b
12	DlvryStat	Delivery Status	R/O	0
11–8	reserved		R/O	0h
7–0	Vector	Vector	R/W	00h

## Field Descriptions

**Mode (Mode)**—Bit 17. This bit is 0 for One-shot and 1 for Periodic.

**Mask (Mask)**—Bit 16. If this bit is set, this LVT Entry will not generate interrupts.

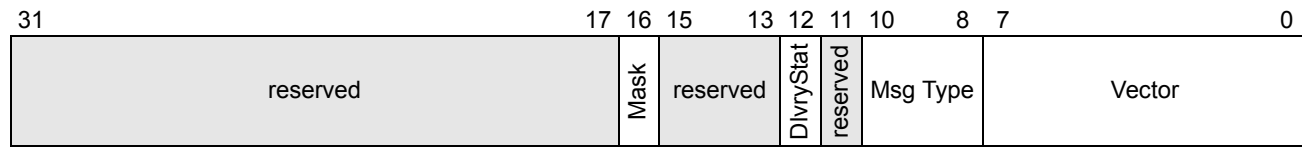
**Delivery Status (DlvryStat)**—Bit 12. This bit is set to indicate that the interrupt has not yet been accepted by the CPU.

**Vector (Vector)**—Bits 7–0. This field contains the vector that will be sent for this interrupt source.

## 8.8.17 Performance Counter Local Vector Table Entry

### PERF\_CNT\_LVT Register

Offset 340h



Bit	Name	Function	R/W	Reset
31–17	reserved		R/O	0000h
16	Mask	Mask	R/W	1
15–13	reserved		R/O	000b
12	DlvryStat	Delivery Status	R/O	0
11	reserved		R/O	0
10–8	MsgType	Message Type	R/W	000b
7–0	Vector	Vector	R/W	00h

### Field Descriptions

**Mask (Mask)**—Bit 16. If this bit is set, this LVT Entry will not generate interrupts.

**Delivery Status (DlvryStat)**—Bit 12. This bit is set to indicate that the interrupt has not yet been accepted by the CPU.

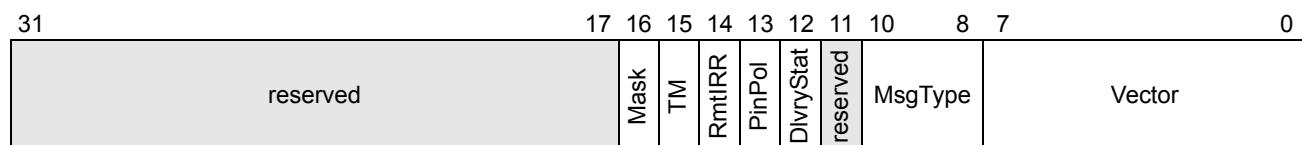
**Message Type (MsgType)**—Bits 10–8. Message Types 000b (Fixed) and 100b (NMI) are legal for Revision D and earlier revisions. Message Types 000b (Fixed), 010b (SMI), and 100b (NMI) are legal for Revision E. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Vector (Vector)**—Bits 7–0. This field contains the vector that will be sent for this interrupt source.

## 8.8.18 Local Interrupt 0 (Legacy INTR) Local Vector Table Entry Register

### LINT0\_LVT Register

Offset 350h



Bit	Name	Function	R/W	Reset
31–17	reserved		R/O	0000h
16	Mask	Mask	R/W	1
15	TM	Trigger Mode	R/W	0



Bit	Name	Function	R/W	Reset
14	RmtIRR	Remote IRR	R/O	0
13	PinPol	Pin Polarity	R/W	0
12	DlvryStat	Delivery Status	R/O	0
11	reserved		R/O	0
10–8	MsgType	Message Type	R/W	000b
7–0	Vector	Vector	R/W	00h

## Field Descriptions

**Mask (Mask)**—Bit 16. If this bit is set, this LVT Entry will not generate interrupts.

**Trigger Mode (TM)**—Bit 15. This bit is set for level-triggered interrupts and clear for edge-triggered interrupts.

**Remote IRR (RmtIRR)**—Bit 14. If trigger mode is level, remote IRR is set when the interrupt has begun service. Remote IRR is cleared when the EOI has occurred.

**Pin Polarity (PinPol)**—Bit 13. This bit is not used because LINT interrupts are delivered by HyperTransport™ messages instead of individual pins.

**Delivery Status (DlvryStat)**—Bit 12. This bit is set to indicate that the interrupt has not yet been accepted by the CPU.

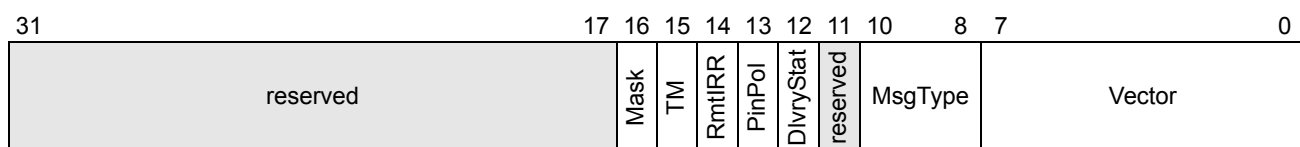
**Message Type (MsgType)**—Bits 10–8. Only Message Types 000b (Fixed), 100b (NMI), and 111b (External Interrupt) are legal.

**Vector (Vector)**—Bits 7–0. This field contains the vector that will be sent for this interrupt source.

## 8.8.19 Local Interrupt 1 (Legacy NMI) Local Vector Table Entry

### LINT1\_LVT Register

Offset 360h



Bit	Name	Function	R/W	Reset
31–17	reserved		R/O	0000h
16	Mask	Mask	R/W	1
15	TM	Trigger Mode	R/W	0
14	RmtIRR	Remote IRR	R/O	0
13	PinPol	Pin Polarity	R/W	0
12	DlvryStat	Delivery Status	R/O	0
11	reserved		R/O	0

Bit	Name	Function	R/W	Reset
10–8	MsgType	Message Type	R/W	000b
7–0	Vector	Vector	R/W	00h

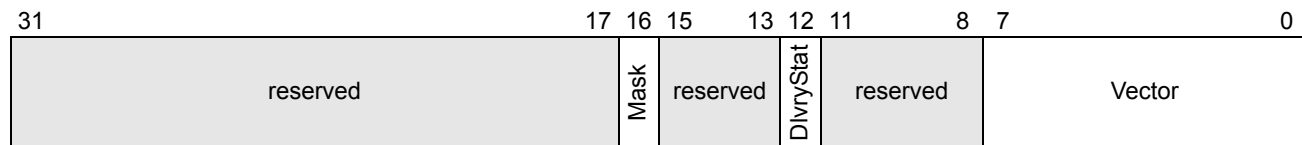
## Field Descriptions

These fields are the same as those for Local Interrupt 0.

### 8.8.20 Error Local Vector Table Entry

#### ERROR\_LVT Register

Offset 370h



Bit	Name	Function	R/W	Reset
31–17	reserved		R/O	0000h
16	Mask	Mask	R/W	1
15–13	reserved		R/O	000b
12	DlvryStat	Delivery Status	R/O	0
11–8	reserved		R/O	0h
7–0	Vector	Vector	R/W	00h

## Field Descriptions

**Mask (Mask)**—Bit 16. If this bit is set, this LVT Entry will not generate interrupts.

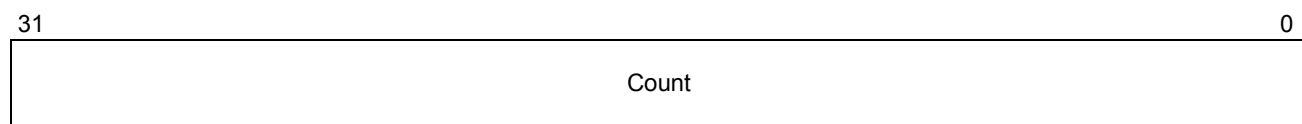
**Delivery Status (DlvryStat)**—Bit 12. This bit is set to indicate that the interrupt has not yet been accepted by the CPU.

**Vector (Vector)**—Bits 7–0. This field contains the vector that will be sent for this interrupt source.

### 8.8.21 Timer Initial Count Register

#### INIT\_CNT Register

Offset 380h



Bit	Name	Function	R/W	Reset
31–0	Count	Initial Count Value	R/W	0000_0000h

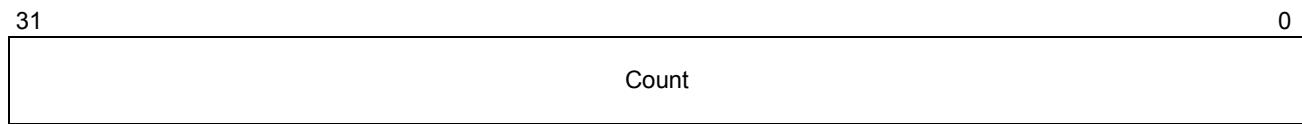
## Field Descriptions

**Count (Count)**—Bits 31–0. This field contains the value copied into the current count register when the timer is loaded or reloaded.

### 8.8.22 Timer Current Count Register

#### CURR\_CNT Register

Offset 390h



Bit	Name	Function	R/W	Reset
31–0	Count	Current Count Value	R/O	0000_0000h

## Field Descriptions

**Count (Count)**—Bits 31–0. This field contains the current value of the counter.

### 8.8.23 Timer Divide Configuration Register

#### TIMER\_DVD\_CFG Register

Offset 3E0h



Bit	Name	Function	R/W	Reset
31–4	reserved		R/O	0000_00h
3	Div[3]	Div[3]	R/W	0
2	reserved		R/O	0
1–0	Div[1–0]	Div[1–0]	R/W	00b

## Field Descriptions

**Div[3] and Div[1–0]**—Bits 3 and 1–0. The Div bits are encoded as follows:

Div[3]	Div[1–0]	Resulting Timer Divide
0	00	2
0	01	4
0	10	8

0	11	16
1	00	32
1	01	64
1	10	128
1	11	1

## 9 Power and Thermal Management

AMD Athlon™ 64 and AMD Opteron™ Processors support ACPI-compliant power management for all classes of systems from notebook PCs to multiprocessor servers. Table 66 lists various levels of power management. Table 67 on page 262 lists ACPI-state support by system class. This chapter covers processor-level power management and processor-specific aspects of system-level power management.

**Table 66. Power Management Categories**

Power Management Category	Comments
System Level	Coarse level power management applied to the entire system, typically after a predefined period of inactivity, or in response to a user action such as pressing the system power button. <ul style="list-style-type: none"> <li>• System Sleep States (ACPI-defined S-states)</li> </ul>
Processor Level	<ul style="list-style-type: none"> <li>• Processor Power States (ACPI-defined C-states)</li> <li>• Processor Performance States (ACPI-defined P-states)</li> <li>• Software Transparent Power Management (Hardware enforced throttling).</li> </ul>
Device Level	<ul style="list-style-type: none"> <li>• Device Power States (ACPI-defined D-states)</li> <li>• Device Performance States</li> <li>• Software Transparent Power Management.</li> </ul>
Bus Level	<ul style="list-style-type: none"> <li>• Bus Power States</li> <li>• Software Transparent Power Management</li> </ul>
Sub-system Level	<ul style="list-style-type: none"> <li>• Typically not used in PC systems, and beyond the scope of this document.</li> </ul>

**Table 67. ACPI-State Support by System Class**

ACPI State Support	Mobile Systems	Uniprocessor Desktop PCs	Multiprocessor Systems
G0/S0/C0: Working	Yes	Yes	Yes
G0/S0/C0: Processor performance state (P-state) transitions under OS control.	Yes	Yes <sup>1</sup>	No
G0/S0/C0: Thermal clock throttling (I/O hub hardware enforced)	Yes	Yes	Revision B:No <sup>1</sup> Revision C:Yes
G0/S0/C0: Thermal clock throttling (operating system enforced)	Yes	Yes	No
G0/S0/C1: Halt	Yes	Yes	Yes
G0/S0/C2: Stop Grant Caches snoopable	Yes <sup>1,2</sup>	No	No
G0/S0/C3: Stop Grant Caches not snoopable	Yes <sup>1,2</sup>	No	No
G1/S1: Stand By (Powered On Suspend)	Yes	Yes	Yes
G1/S3: Stand By (Suspend to RAM)	Yes	Yes	Revision B:No <sup>1</sup> Revision C:Yes
G1/S4: Hibernate (Suspend to Disk)	Yes	Yes	Yes
G2/S5: Shut Down, Turn Off (Soft Off)	Yes	Yes	Yes
G3 Mechanical Off	Yes	Yes	Yes
<b>Notes:</b> <ol style="list-style-type: none"> <li>1. Not supported by some revisions of the silicon. Refer to Revision Guide for AMD Athlon™ 64 and AMD Opteron™ Processors, order# 25759.</li> <li>2. See "Advanced Programmable Interrupt Controller (APIC)" on page 239.</li> </ol>			

## 9.1 Stop Grant

The processor and chipset use HyperTransport™ technology STPCLK and Stop Grant system management messages to sequence into all power management states except for Halt. These messages carry a 3-bit System Management Action Field (SMAF) to differentiate the various reasons for placing the processor into the Stop Grant state. Table 68 maps Stop Grant SMAF codes to ACPI states and actions that the BIOS is required to configure the processor to take for each ACPI state.

**Table 68. Required SMAF Code to Stop Grant Mapping**

<b>Reason for STPCLK Message</b>	<b>SMAF Field of STPCLK and Stop Grant Messages</b>	<b>Mechanism that Forces the I/O Hub to Send the STPCLK Message</b>	<b>Processors Programmed Response after Entering the Stop Grant State (See Table 78 Power Management Control Registers.)</b>
C2 Stop Grant Caches Snooperable	000b	Sent in response to a read of the ACPI-defined P_LVL2 register.	Ramp the CPU clock grid down when no probes need to be serviced. (CPU Low Power Enable)
C3 Stop Grant Caches not Snooperable. Used only by mobile systems.	001b	Sent either in response to a read of the ACPI-defined P_LVL3 register or in response to a write to the Link Frequency Change and Resize LDTSTOP_L Command register in the I/O hub.	Ramp the CPU clock grid frequency down. (CPU Low Power Enable)  After LDTSTOP_L is asserted, place memory into self-refresh and ramp down the processor host bridge and memory controller clock grid. (Northbridge Low Power Enable)
VID/FID Change Or Link Width/Frequency changes	010b	Sent either in response to a VID/FID Change special cycle from the processor or in response to a write to the Link Frequency Change and Resize LDTSTOP_L Command register.	After LDTSTOP_L assertion, place memory into self-refresh, ramp the processor clock grids down, then drive new FID values to PLL. (CPU Low Power Enable, Northbridge Low Power Enable, FID/VID Change Enable)
S1 Sleep state	011b	Sent in response to writing the S1 value to the SLP_TYP[2:0] field and setting the SLP_EN bit of the ACPI-defined PM1 control register in the I/O hub.	Same response as C3.
S3 Sleep state	100b	Sent in response to writing the S3 value to the SLP_TYP[2:0] field of the PM1 control register.	Same response as C3. Additionally, after LDTSTOP_L has been asserted, the I/O hub will power off the main power planes.
Throttling	101b	Occurs based upon hardware initiated thermal throttling or ACPI-controlled throttling.	Ramp down the CPU grid by the programmed amount. (CPU Low Power Enable)
S4/S5	110b	Sent in response to writing the S4 or S5 value to the SLP_TYP[2:0] field of the PM1 control register.	Same response as S3 for processor. Additionally, all power will be removed from processor during S4 and S5.

**Table 68. Required SMAF Code to Stop Grant Mapping (Continued)**

Reason for STPCLK Message	SMAF Field of STPCLK and Stop Grant Messages	Mechanism that Forces the I/O Hub to Send the STPCLK Message	Processors Programmed Response after Entering the Stop Grant State (See Table 78 Power Management Control Registers.)
Reserved for use by processor	111b	No I/O hub STPCLK message uses this SMAF code. The power management register field corresponding to this SMAF code is used by the processor in response to executing the Halt instruction.	Same response as C2, except a Halt special cycle is broadcast.

## 9.2 C-States

As Table 66 on page 261 indicates, the processor supports ACPI-defined processor power states, which are referred to as C-states. Table 67 on page 262 indicates which C-states are supported by system class. The operating system will place the processor into C-states when the processor is idle an operating-system-determined percentage of the time.

### 9.2.1 C1 Halt State

C1 is the Halt state. C1 is entered after the HLT instruction has been executed. The BIOS configures the processor to enter a low-power state during C1, in which the CPU core clock grid is ramped down from its operating frequency. See Table 78 on page 289.

### 9.2.2 C2 and C3

C2 is the Stop Grant state in which the processor caches can be snooped. When the processor is in the Stop Grant state and there is no probe activity to the processor's caches, the CPU core clock grid is ramped down from its operating frequency. For more information, see Table 78 on page 289.

C3 is the Stop Grant state in which the processor's caches cannot be snooped. The chipset asserts LDTSTOP\_L during the C3 state. The chipset may require BIOS to enable LDTSTOP\_L assertion for the C3 state. When the processor is in the Stop Grant state, the CPU core clock grid is ramped down from its operating frequency. After LDTSTOP\_L assertion, the processor's system memory is placed into self-refresh mode and the host bridge/memory controller clock grid is ramped down from their operational frequency. For more information, see Table 78 on page 289.

### 9.2.3 C3 and AltVID

For mobile processors that are revision E and later revisions that use discrete graphics, the hardware can be programmed to drive an alternate VID code during the C3 processor power state. The purpose



of the AltVID is to reduce the processor voltage to the minimum operational level while the processor is in the C3 state. AltVID settings are documented in the power and thermal data sheet. BIOS must use the parameters documented in Section 9.5.1.2 to match a processor with the appropriate AltVID setting. See Table 78 on page 289 for AltVID configuration register settings.

## 9.3 Throttling

The BIOS must declare a DUTY\_WIDTH value of zero in the Fixed ACPI Description table if the system does not support the \_PSV object as part of a processor ACPI thermal zone. Throttling is not supported by some revisions of the silicon. Refer to *AMD Revision Guide for AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25759.

Chipsets must not be configured to assert LDTSTOP\_L during throttling.

## 9.4 Processor ACPI Thermal Zone

This section will be part of a future revision of this document.

## 9.5 Processor Performance States

Processor performance states (P-states) are valid operating combinations of processor core voltage and frequency. The hardware supporting P-state transitions in AMD processors is referred to as AMD PowerNow!™ technology for mobile systems and AMD Cool'n'Quiet™ technology for desktop systems. In this document all descriptions of AMD PowerNow!™ technology, software and driver apply to AMD Cool'n'Quiet™ technology, software, and driver.

- For operating systems without native support for P-state transitions (legacy operating systems) using Revision C and subsequent processors, AMD PowerNow! software is used to perform P-state transitions. In this case, a BIOS generated Performance State Block (PSB) is used by AMD PowerNow! software to determine what P-states are supported by the processor in the system.
- For operating systems that have native support for P-state transitions using Revision C and subsequent processors, a processor-specific driver is used by the operating system to make P-state transitions. In this case, BIOS-provided ACPI-defined P-state objects inform the operating system that P-state transitions are supported by the system, which P-states are supported, and when given P-states are available for use by the operating system.

The power and thermal data sheet specifies the valid P-states for each processor. The BIOS vendor must only use P-states defined in that document for a given processor when building the ACPI P-state objects and PSB for use by higher-level software. BIOS may not be required to implement all of the valid P-states for a processor. Any optional P-states will be clearly noted in the power and thermal data sheet.

Processors that support P-state transitions provide MSR C001\_0041h (FIDVID\_CTL) and MSR C001\_0042h (FIDVID\_STATUS) for initiating P-state transitions and verifying that they are complete. If these two MSRs are accessed in a processor that does not support P-state transitions, then a general protection fault occurs.

Enabling P-state transitions in systems that use external clock buffers for unbuffered 3-DIMM configurations is not recommended.

Chipsets provide support for P-state transitions. Refer to the chipset documentation for information on chipset configuration and P-state-related considerations.

## **9.5.1 BIOS Requirements for P-State Transitions**

P-state transitions can be used only if they are supported by the processor and by the system.

BIOS is required to:

- Determine that the processor supports P-state transitions.
- Have a valid set of P-states for the processor, based on the power and thermal data sheet.
- Take chipset or system limitations into account before providing the ACPI-defined P-state objects described in 9.6 on page 277, or the PSB described in 9.7 on page 287 of this document.
- If P-states are supported and a Revision C or higher revision processor is present, BIOS is required to provide both:
  - ACPI-defined P-state objects for operating systems that support native P-state control.
  - a PSB in support of AMD PowerNow! software with legacy operating systems.

The BIOS must not provide the ACPI P-state objects or the PSB if:

- The processor does not support P-state transitions.
- The processor is Revision B.
- The chipset or system does not support P-state transitions.
- The BIOS does not have a set of valid P-states derived from the power and thermal data sheet.

### **9.5.1.1 Step 1: Determine Processor Support for P-State Transitions**

The BIOS determines if a processor supports P-state transitions by executing the CPUID extended function 8000\_0007h.

If the value returned in EDX[2:1] is:

- 11b, then the part is capable of performing P-state transitions.
- 00b, then the part is *not* capable of performing P-state transitions.

Refer to the following documents regarding CUID and determining which processor is in the system:

- *AMD Processor Recognition Application Note*, order# 20734
- *CUID Guide for the AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25481

### 9.5.1.2 Step 2: Match the Processor to a Valid Set of P-States

If CUID extended function 8000\_0007h returned EDX[2:1] = 11b, the BIOS must correlate the processor in the system to a valid set of P-states from the power and thermal data sheet. Each processor is uniquely identified by the following:

- CUID extended function 8000\_0001h, EAX register:
  - Family number and extended family number (when family number field = 1111b).
  - Model number and extended model number (when family number field = 1111b).
  - Stepping (Revision).
- CUID extended function 8000\_0001h, EBX register (BrandID information):
  - VID code mapping defined in Table 74, Voltage column should be used.
- FIDVID\_STATUS MSR (C001\_0042h):
  - MaxVID<sup>1</sup>: The VID requested VDD supply level (VID\_VDD) voltage for the maximum P-state is the voltage selected by MaxVID minus the ramp voltage offset (RVO).
  - StartVID: This corresponds to the VID\_VDD voltage at which the processor starts operation after a cold reset.
  - MaxFID: This corresponds to the frequency that the processor operates at when in the maximum performance state.
  - StartFID: This corresponds to the frequency that the processor starts operation at after a cold reset.

These parameters are used by the BIOS to match the processor in the system to a valid set of P-states (voltage and frequency combinations<sup>1</sup>) and AltVID setting (for mobile systems only, see Section 9.2.3) listed in the power and thermal data sheet.

Notes:

1. For revision E and later, some product offerings may have more than one VID\_VDD voltage option for the maximum P-state specified in the power and thermal data sheet. BIOS can correlate MaxVID to the appropriate VID\_VDD voltage for the maximum P-state of a specific device using the formula  $VID\_VDD = MaxVID - RVO$ .

### **9.5.1.3 Step 3: Determine System Support for P-State Transitions**

The BIOS software must also determine whether the system supports P-state transitions. The BIOS must take into account chipset/motherboard components and potential system side effects associated with P-state transitions that could preclude the use of P-state transitions.

## **9.5.2 BIOS Requirements for P-State Transitions in Systems Using Dual Core Processors**

- For systems with a single processor and unbuffered DIMMs or SODIMMs, BIOS configures the processor in the same way as described in Section 9.5.1. with the exception that one set of the `_PCT`, `_PSS`, and `_PPC` ACPI objects must be declared for each processor core.

## **9.5.3 BIOS-Initiated P-State Transitions**

Processors that do not support P-state transitions and processors intended for desktop and server systems power up to the maximum P-state. Processors intended for mobile systems power up in the minimum P-state. BIOS can transition mobile processors from the minimum P-state to a higher P-state during the POST routine. If BIOS performs a P-state transition it must follow the P-state transition algorithm defined in 9.5.6 on page 269. The BIOS never initiates P-state transitions after passing control to the operating system.

## **9.5.4 BIOS Support for Operating System/CPU Driver-Initiated P-State Transitions**

Operating systems that have native control for processor P-states exclusively use the presence of BIOS provided ACPI 2.0 defined processor P-state objects to determine if processor P-states are supported in a system. If the ACPI objects do not exist, then the processor driver is not loaded.

In operating systems that do not have native support for processor P-state transitions, an AMD defined PSB provided by the BIOS informs AMD PowerNow! technology software that P-state transitions are supported.

Once it has been determined that a processor and system support P-state transitions, the BIOS must provide both:

- ACPI-2.0-defined processor P-state objects for operating systems with native P-state support. See Section 9.6 on page 277.
- AMD-defined PSB for use with AMD PowerNow! software and operating systems that do not have native P-state support. See Section 9.7 on page 287.

### 9.5.5 Processor Driver Requirements

The processor driver that supports native operating system policy and control of P-state objects is required to use the ACPI 2.0 P-state objects as defined in this document. The processor driver is not permitted to use the legacy PSB tables described in section 9.7 on page 287.

### 9.5.6 P-State Transition Sequence

This section describes the P-state transition algorithm for the AMD Athlon™ 64 and AMD Opteron™ processors.

#### 9.5.6.1 FID to VCO Frequency Relationship

Table 69 and Table 70 provide the core frequency-to-VCO frequency relationship for AMD Athlon™ 64 and AMD Opteron™ processors. These processors do not support frequency transitions in VCO frequency steps greater than 200 MHz. The processor driver, and the AMD PowerNow! driver use these tables when making P-state transitions.

Only one P-state is allowed in Table 69. The one P-state consists of a frequency from the “Minimum Core Frequency” column in Table 69 and is the minimum P-State supported by the processor. For processors that support P-state transitions, the power and thermal data sheet has a table of valid P-states based on ordering part number that dictates which frequency in Table 69 is used for the minimum P-state supported by the processor.

Table 69 additionally defines Portal Core Frequencies. The “Portal Core Frequencies in Table 70” column of Table 69 defines:

- The core frequencies in Table 70 to which direct transitions can be made from the minimum core frequency in Table 69.
- The core frequencies in Table 70 that support transitions to the minimum core frequency in Table 69.

The processor supports direct transitions between the minimum core frequency in Table 69 and any of the core frequencies from Table 70 listed in the Portal Core Frequencies column associated with the minimum core frequency.

If the minimum P-state from the low table (Table 69) has a VCO frequency  $> 1600$  MHz, then the VCO/core frequency of all P-states in the high table (Table 70) must be  $\geq$  the VCO frequency of the minimum P-state minus 200 MHz.

#### Example 1:

Given a processor with a minimum core frequency of 800 MHz and a maximum core frequency of 2000 MHz, to transition from 800 MHz to 2000 MHz, the processor driver will (in the order specified):

1. Transition the processor from the 800 MHz core frequency to 1800 MHz core frequency. 1800 MHz is a portal frequency to and from 800 MHz as defined in Table 69, because the VCO frequency change from an 800 MHz core frequency to an 1800 MHz core frequency is  $\leq 200$  MHz.
2. Transition the processor core frequency from 1800 MHz to 2000 MHz (VCO frequency change is  $\leq 200$  MHz according to Table 70).

**Note:** To simplify this example, the voltage transitions and isochronous relief times are not described in this example.

### Example 2:

Given a processor with a minimum core frequency of 1400 MHz and a maximum core frequency of 3400 MHz, to transition from 1400 MHz to 3400 MHz, the processor driver will (in the order specified):

1. Transition the processor from the core frequency of 1400 MHz to a core frequency of 3000 MHz. 3000 MHz is a portal frequency to and from 1400 MHz as defined in Table 69, because the VCO frequency change from 1400 MHz core frequency to 3000 MHz core frequency is  $\leq 200$  MHz.
2. Transition the processor core frequency from 3000 MHz to 3200 MHz (VCO frequency change is  $\leq 200$  MHz per Table 70).
3. Transition the processor core frequency from 3200 MHz to 3400 MHz (VCO frequency change is  $\leq 200$  MHz per Table 70).

**Note:** Note: to simplify this example, the voltage transitions and isochronous relief times are not described in this example.

**Table 69. Low FID Frequency Table ( $< 1600$  MHz)**

FID[5:0]	Minimum Core Frequency MHz	VCO Frequency MHz	Portal Core Frequencies in Table 70
000000b	800	1600	1600, 1800
000010b	1000	2000	1800, 2000, 2200
000100b	1200	2400	2200, 2400, 2600
000110b	1400	2800	2600, 2800, 3000

**Table 70. High FID Frequency Table ( $\geq 1600$  MHz)**

FID[5:0]	Core Frequency MHz	VCO Frequency MHz
001000b	1600	1600
001010b	1800	1800
001100b	2000	2000

**Table 70. High FID Frequency Table (>= 1600 MHz) (Continued)**

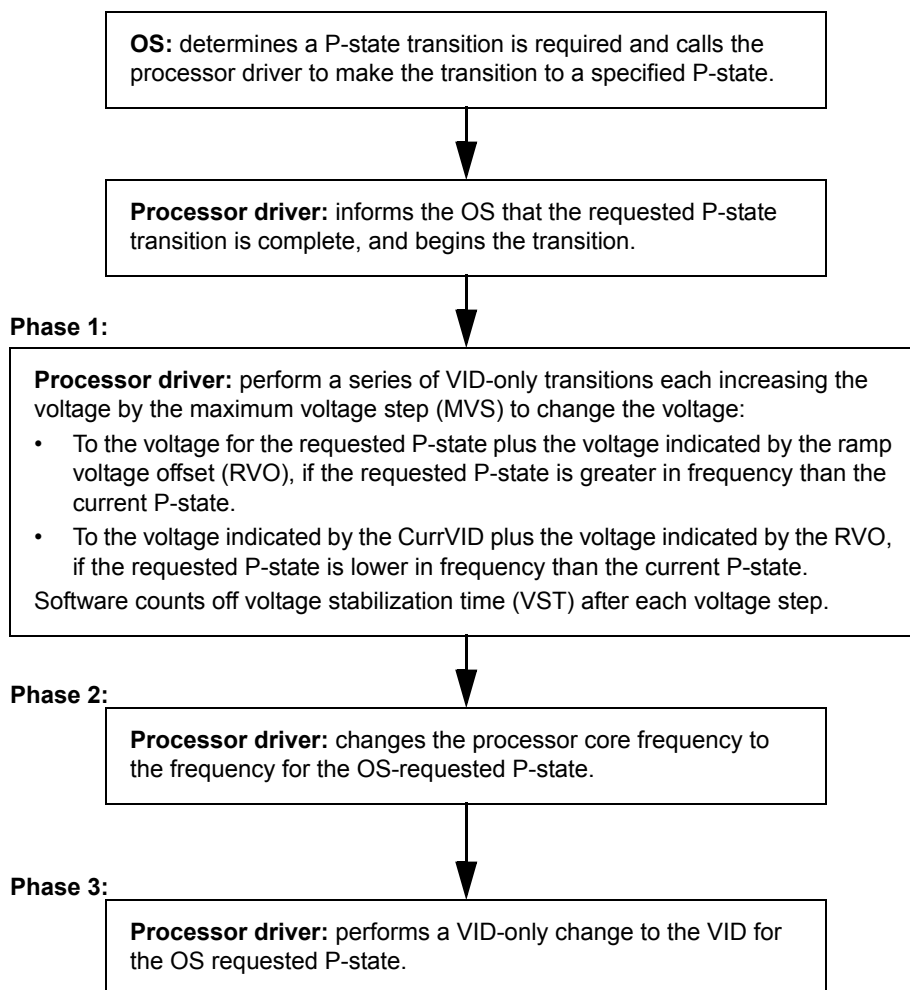
FID[5:0]	Core Frequency MHz	VCO Frequency MHz
001110b	2200	2200
010000b	2400	2400
010010b	2600	2600
010100b	2800	2800
010110b	3000	3000
011000b	3200	3200
011010b	3400	3400
011100b	3600	3600
011110b	3800	3800
100000b	4000	4000
100010b	4200	4200
100100b	4400	4400
100110b	4600	4600
101000b	4800	4800
101010b	5000	5000

### 9.5.6.2 P-state Transition Algorithm

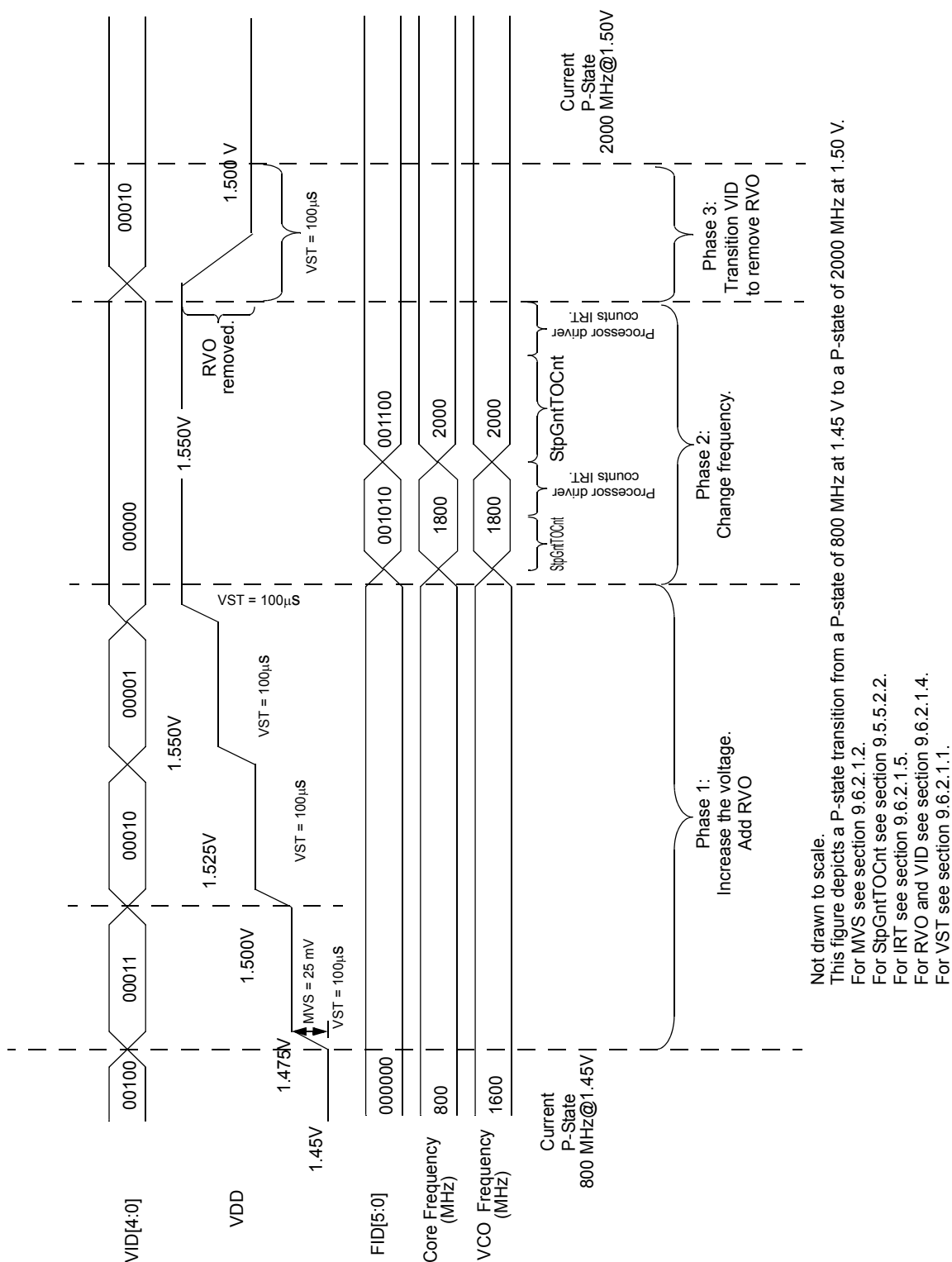
The P-state transition algorithm has three phases. During phase 1 the processor voltage is transitioned to the level required to support frequency transitions. During phase 2 the processor frequency is transitioned to frequency associated with the OS-requested P-state. During phase 3 the processor voltage is transitioned to the voltage associated with the OS-requested P-state.

Figure 6 shows the high-level P-state transition flow. Figure 7 is a high-level timing diagram depicting voltage and frequency stepping associated with each phase of a P-state transition. Figure 8 shows the core voltage transition flow for phase 1 of a P-state transition. Figure 9 shows the core frequency transition flow for phase 2 of a P-state transition. Figure 10 shows the core voltage transition flow for phase 3 of a P-state transition.

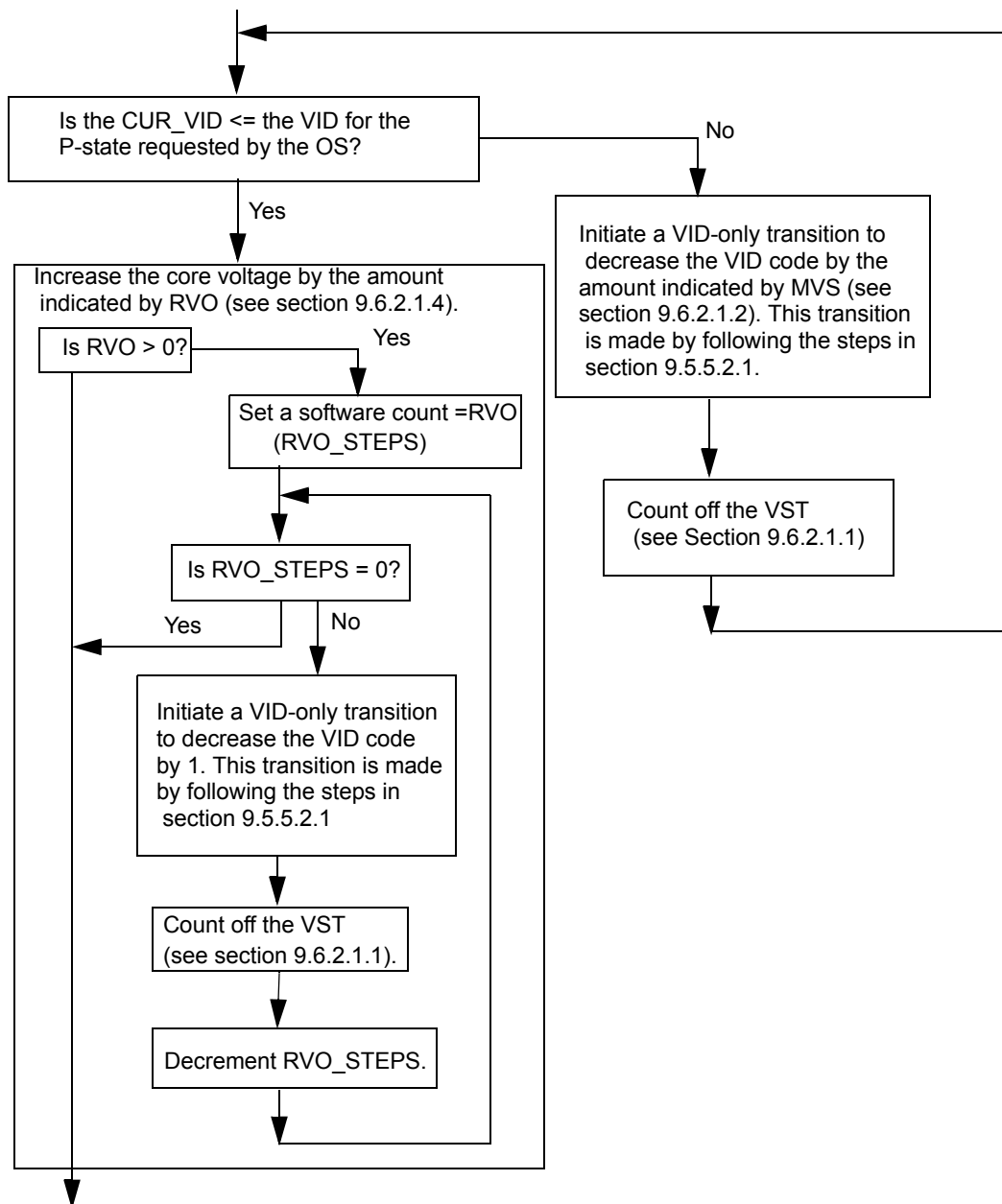
The algorithm shown in Figure 6 through Figure 10 describes the 3-phases of a P-state transition in the context of Windows XP and the associated processor driver, but the algorithm also applies to the AMD PowerNow! driver and any other software that performs P-state transitions (such as a debug tool).

**Figure 6. High-Level P-state Transition Flow**

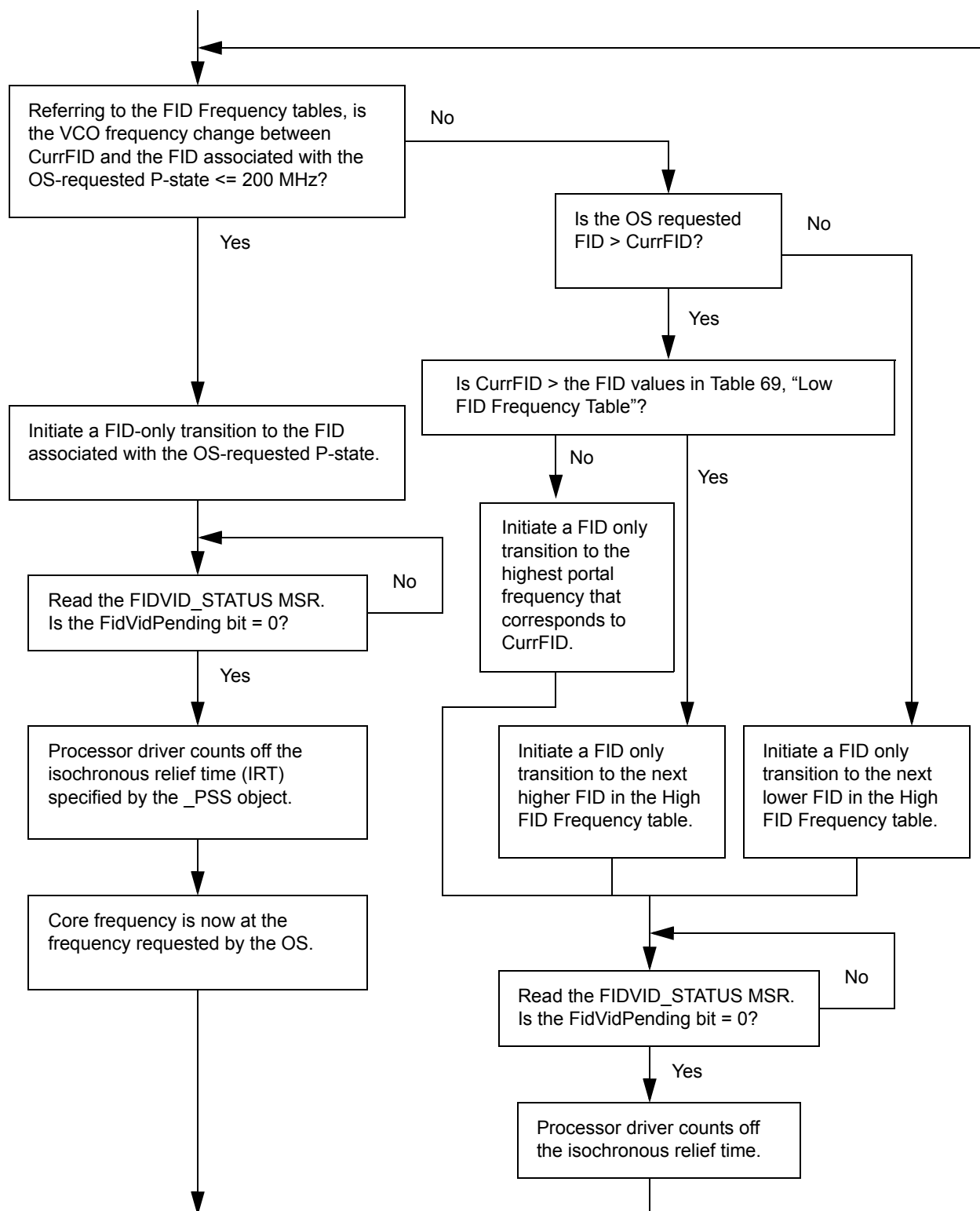


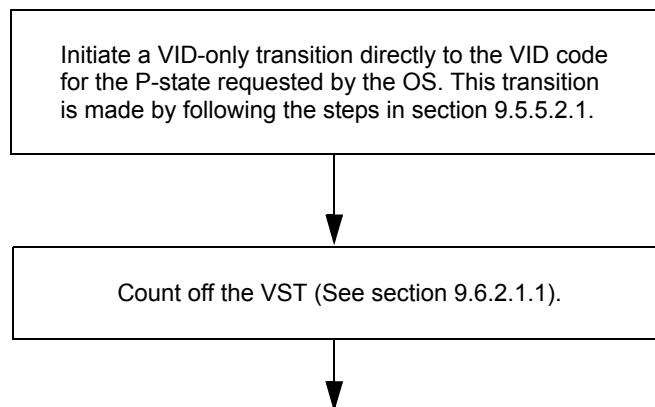


**Figure 7. Example P-State Transition Timing Diagram**



**Figure 8. Phase 1: Core Voltage Transition Flow**

**Figure 9. Phase 2: Core Frequency Transition Flow**



**Figure 10. Phase 3: Core Voltage Transition Flow**

#### 9.5.6.2.1 Changing the VID

**Note:** Software must hold the FID constant when changing the VID.

To change the processor voltage:

1. Write the following values to FIDVID\_CTL (MSR C001\_0041h):
  - NewVID field (bits 12–8) with the VID code associated with the target voltage
  - NewFID field (bits 5–0) with the CurrFID value indicated in the FIDVID\_STATUS MSR
  - StpGntTOCnt field (bits 51–32) with 1h, which corresponds to 5 nsec. For Revision B processors, this setting results in voltage transitions causing the minimum possible memory access latency. This field has no effect on VID changes for Revision C AMD Athlon™ 64 and AMD Opteron™ Processors, since the processor drives the VID[4:0] in response to the MSR write without the processor first being placed into the Stop Grant state.
  - InitFidVid bit (bit 16) set to 1. Setting this bit initiates the VID change.
  - Clear all other bits to 0.
2. Loop on reading the FidVidPending bit (bit 31) of FIDVID\_STATUS (MSR C001\_0042h) until the bit returns 0. The FidVidPending bit stays set to 1 until the new VID code has been driven to the voltage regulator.

#### 9.5.6.2.2 Changing the FID

The AMD Athlon™ 64 and AMD Opteron™ Processors support direct FID transitions only between FIDs that represent a VCO frequency change less than or equal to 200 MHz. To change between FIDs that represent a VCO frequency change greater than 200 MHz, software must make multiple transitions each less than or equal to a 200 MHz change in VCO frequency. For information about allowable FID transitions, see Tables 69 and 70 on page 270.

**Note:** Software must hold the VID constant when changing the FID.

To change the core frequency:

1. Write the following values to FIDVID\_CTL (MSR C001\_0041h):
  - NewVID field (bits 12–8) with the CurrVID value reported in the FIDVID\_STATUS MSR.
  - NewFID field (bits 5–0) with the FID code associated with the target frequency.
  - StpGntTOCnt field (bits 51–32) with a value corresponding to the processor PLL lock time. The PLL lock time is specified by the processor data sheet and refers to all components of PLL lock including frequency lock, phase lock, and settling time. PLL lock time is communicated to the processor driver through the PLL\_LOCK\_TIME field of the \_PSS object. PLL lock time is in microseconds. To translate the PLL lock time to an StpGntTOCnt value, multiply PLL\_LOCK\_TIME by 1000 to get nanoseconds, then divide by 5 which is the clock period of the counter in ns. Therefore, StpGntTOCnt value = PLL\_LOCK\_TIME \* 1000 / 5. For example, a PLL lock time of 2  $\mu$ s results in an StpGntTOCnt value of 400 decimal and 190h.
  - InitFidVid bit (bit 16) set to 1. Setting this bit initiates the P-state transition.
  - Clear all other bits to 0.
2. Loop on reading the FidVidPending bit (bit 31) of FIDVID\_STATUS (MSR C001\_0042h) until the bit returns 0. The FidVidPending bit stays set to 1 until the new FID code is in effect.

Figure 9 on page 275 illustrates the transition flow for core frequencies, including the requirement to wait the isochronous relief time between FID steps.

### 9.5.6.3 Dual Core P-state Transition Requirements

For dual core processors, it is the processor driver's responsibility to coordinate operating systems P-state transition requests for the individual cores. The cores in a dual core processor share a common clock grid and voltage plane, and therefore do not support independent P-states even though the operating system views the cores as independent processors, and a set of ACPI P-state objects is declared to the operating system for each core. The processor driver enforces a "highest requested P-state" policy. Both cores will be in the highest P-state requested for either core. The processor driver keeps track of P-state transition requests and only reduces the P-state of either core after the operating system has requested that the P-state of both cores be reduced. When the operating system requests that the P-state of either core be increased, the processor driver increases the P-state of both cores.

## 9.6 ACPI 2.0 Processor P-State Objects

For systems that support P-state transitions, the BIOS must provide the following ACPI 2.0 P-state objects to support operating systems that provide native support for processor P-state transitions. These subsections discuss the specific implementation of these objects. For dual core processors, one set of ACPI P-state objects is declared for each core.

### 9.6.1 \_PCT (Performance Control)

The \_PCT object is generically described in section 8.3.3.1 of the ACPI 2.0 specification. The \_PCT, \_PSS, and \_PPC objects (defined in the next sections) are all placed after the Processor object in the \\_PR name space for operating systems that have native support for processor P-states. This section provides BIOS specifics regarding the use of \_PCT.

The \_PCT object specifies the control register and status register used to initiate and verify P-state transitions. The processor's P-state transition protocol is abstracted from the operating system by the processor driver, so this object does not declare the MSRs used for FID\_Change protocol, but rather declares these two registers to be *functional fixed hardware*. Functional fixed hardware means that the mechanism is specific to the processor, and the processor driver knows how to initiate and verify P-state transitions because the processor driver is written by the processor vendor or written based upon documentation supplied to the operating system vendor by the processor vendor.

The following is a sample \_PCT object. Sections referenced are in the ACPI 2.0 specification and ACPI 2.0 Errata Document, revision 1.4.

```
Name (_PCT, Package (2)          // Performance Control Object
{
    //      ResourceTemplate () {Register(FFixedHW, 0, 0, 0)} // Control
    Buffer () {
        0x82,          // B0-Generic Register Descriptor (Sec. 6.4.3.7)
        0xC,0,          // B1:2-length (from _ASI through _ADR fields)
        0x7F,          // B3-Address space ID, _ASI, SystemIO
        0,              // B4-Register Bit Width, _RBW
        0,              // B5-Register Bit Offset, _RBO
        0,              // B6-Reserved. Must be 0.
        0,0,0,0,0,0,0,0, // B7:14-register address, _ADR (64bits)
        0x79,0},        // B15:16-End Tag (Section 6.4.2.8)

    //      ResourceTemplate () {Register(FFixedHW, 0, 0, 0)} // Status
    Buffer () {
        0x82,          // B0-Generic Register Descriptor (Section
6.4.3.7)
        0xC,0,          // B1:2-length (2 bytes)
        0x7F,          // B3-Address space ID, _ASI, SystemIO
        0,              // B4-Register Bit Width, _RBW
        0,              // B5-Register Bit Offset, _RBO
        0,              // B6-Reserved. Must be 0.
        0,0,0,0,0,0,0,0, // B7:14-register address, _ADR (64bits)
        0x79,0},        // B15:16-End Tag (Section 6.4.2.8)
    }) // End of _PCT object
```

### 9.6.2 \_PSS (Performance-Supported States)

The \_PSS object is generically described in Section 8.3.3.2 of the ACPI 2.0 specification. This object follows the \_PCT object in the \\_PR name space in support of operating systems with native support for processor P-states. This section provides the BIOS specifics regarding use of \_PSS.

The `_PSS` object specifies the performance states that are supported by the system.

Based upon the maximum voltage and frequency supported by the processor and system requirements, the BIOS will build the `_PSS` object to specify the performance states supported by the system.

The BIOS creates the `_PSS` object based on the table of valid P-states in the power and thermal data sheet. The BIOS is required to maintain a set of Performance State Tables (PSTs) based on the power and thermal data sheet for all processors capable of P-state transitions. Only the PST entries that correspond to the processor found in the specific system by the BIOS during the POST routine are used by the BIOS to create the `_PSS` object for that system. The `_PSS` object could have as many performance states as the power and thermal data sheet indicates are supported. For example, the `_PSS` object could provide performance state definitions P0–P2:

```
P0: 2400 MHz at 1.20 V
P1: 1600 MHz at 1.10 V
P2: 800 MHz at 1.00 V
```

The generic `_PSS` object description has the following format:

```
Name ( _PSS, Package()
{
    // Field Name Field Type
Package () // Performance State 0 Definition - P0
{
    CoreFreq,      // DWordConst
    Power,         // DWordConst
    TransitionLatency, // DWordConst
    BusMasterLatency, // DWordConst
    Control,       // DWordConst
    Status         // DWordConst
},
Package ()        // Performance State n Definition - Pn
{
    CoreFreq,      // DWordConst
    Power,         // DWordConst
    TransitionLatency, // DWordConst
    BusMasterLatency, // DWordConst
    Control,       // DWordConst
    Status         // DWordConst
}
} )                // End of _PSS object
```

Each performance state entry contains six data fields as follows:

- **CoreFreq** is the core CPU operating frequency (in MHz).

- **Power** is the typical power dissipation (in milliWatts). The BIOS must fill out this field for the maximum performance state as specified in the power and thermal data sheet. Estimates for power consumption for lower P-states are acceptable.
- **TransitionLatency** is the worst-case latency, in microseconds, that the CPU is unavailable during a transition from any performance state to this performance state. During a P-state transition, the CPU is unavailable for no more than 6  $\mu$ s for each frequency step. While the total P-state transition could take up to 2 ms, the operating system is not blocked during the transition. Therefore the recommended value for this field is 100  $\mu$ s.
- **BusMasterLatency** is the worst-case latency, in microseconds, that Bus Masters are prevented from accessing memory during a transition from any performance state to this performance state. This value is estimated at 7  $\mu$ s.
- **Control** is the value to be written to the \_PSS Control Field in order to initiate a transition to the performance state. The BIOS must fill this out as described on page 280.
- **Status** is the value that the processor driver can compare to a value read from the \_PSS Status Field to ensure that the transition to the performance state was successful. The BIOS must fill this out as described on page 284.

### 9.6.2.1 \_PSS Control Field

The control field of each performance state definition within the \_PSS object contains the information that the processor driver uses. This includes NewVID and NewFID values that must be written into the FIDVID\_CTL register (MSR C001\_0041) to initiate transitions between P-states. \_PSS Control field has the following format:

31	30	29	28	27	26		20	19	18	17		11	10		6	5		0
IRT	RVO	$\frac{\infty}{0}$				PLL_LOCK_TIME	MVS				VST				NewVID			NewFID

Bits	Field	Description
31–30	IRT	Isochronous Relief Time
29–28	RVO	Ramp Voltage Offset
27	Reserved	Must be declared as zero.
26–20	PLL_LOCK_TIME	Phase-Locked Loop Time
19–18	MVS	Maximum Voltage Step
17–11	VST	Voltage Stabilization Time
10–6	NewVID	The VID value associated with the OS-requested P-state
5–0	NewFID	The FID value associated with the OS-requested P-state

The following sections describe the sub-fields of the \_PSS control field.



### 9.6.2.1.1 Voltage Stabilization Time

The Voltage Stabilization Time (VST) defines the number of microseconds (in 20  $\mu$ s increments) that are required for the voltage to increase by the amount specified by the MVS field when the VID outputs of the processor transition. The processor driver counts VST between VID steps that increase the processor's core voltage, not between steps that decrease it. Table 71 shows the actual stabilization time for several VST values.

**Table 71. Sample VST Values**

VST (Bits 17–11)	Stabilization Time
7'h00	0 $\mu$ s
7'h01	20 $\mu$ s
...	
7'h05	100 $\mu$ s (BIOS default)
...	
7'h64	2000 $\mu$ s
...	
7'h7e	2520 $\mu$ s
7'h7f	2540 $\mu$ s

### 9.6.2.1.2 Maximum Voltage Step

The Maximum Voltage Step (MVS) defines the maximum voltage increment the processor driver can use when changing from a lower voltage to a higher voltage. For the processor driver, when increasing core voltage, the next VID = CurrVID -  $2^{\text{MVS}}$ . Table 72 on page 282 shows the values and increments for this field.

For example, if the voltage is being increased, and

- the current VID is 00100 (1.45 V),
- the OS-requested target VID is 00000 (1.55 V),
- and MVS is 0 (which selects 25mV voltage steps).

Then, the processor driver calculates the first VID to which to transition as follows:

$$\text{Next\_VID} = \text{CurrVID} - 2^{\text{MVS}} = 00100 - 2^0 = 00100 - 1 = 00011 = 1.475 \text{ V}$$

Next the processor driver transitions the VID to select 1.475 V.

After the VST is counted off, the processor driver calculates the next VID to which to transition as follows:

$$\text{Next\_VID} = \text{CurrVID} - 2^{\text{MVS}} = 00011 - 2^0 = 00010 = 1.500 \text{ V}$$

This process iterates until the CurrVID is equal to the OS-requested target VID minus the VID associated with the ramp voltage offset. After each VID transition the processor driver counts off the voltage stabilization time before calculating the Next\_VID.

**Table 72. MVS Values**

MVS (Bits 19–18)	Increment
00	25 mV (BIOS default. This is the required value that must be used.)
01	50 mV (Reserved for future use)
10	100 mV (Reserved for future use)
11	200 mV (Reserved for future use)

**9.6.2.1.3 PLL Lock Time**

The 7-bit binary PLL\_LOCK\_TIME value defines the time required for the processor PLLs to lock in microseconds. The PLL\_LOCK\_TIME value is specified in the processor data sheet.

**9.6.2.1.4 Ramp Voltage Offset**

The Ramp Voltage Offset (RVO) is the offset voltage applied to the VID requested VDD supply level (VID\_VDD) voltage when performing P-state transitions. The RVO is specified in the processor data sheet. Table 73 lists the RVO values and their corresponding offset voltages.

**Table 73. RVO Values**

RVO (Bits 29–28)	Offset Voltage	Description
00	0 mV	Target P-state VID needs no adjustment for RVO. (BIOS default for Multiprocessor Systems only. This is the required value.)
01	25 mV	Decrement the target P-state VID by 1 to add the RVO.
10	50 mV	Decrement the target P-state VID by 2 to add the RVO. (BIOS default. This is the required value.)
11	75 mV	Decrement the target P-state VID by 3 to add the RVO.

For the processor driver, RVO is in VID increments. To increase the processor voltage the processor driver must subtract the RVO field from the VID. The MVS field dictates the increments in which RVO can be subtracted from VID.

For example, consider the case when the VID is 00100b (1.45 V), RVO is 10b (50 mV), MVS is 00b (25 mV), and VST is 7'h05 (100  $\mu$ s). Based upon these parameters, the voltage must be increased by 50 mV in two steps of 25 mV with a 100  $\mu$ s delay after each VID change. The steps to apply the RVO are:

1. The VID code is reduced to 00011b. This transitions the voltage from 1.45 V to 1.475 V.

2. The processor driver waits 100  $\mu$ s as specified by VST.
3. The VID code is reduced to 00010b. This transitions the voltage from 1.475 V to 1.5 V.
4. The processor driver waits 100  $\mu$ s as specified by VST.

Table 74 lists the VID codes and their corresponding voltages for the AMD Athlon™ 64 and AMD Opteron™ processors.

**Table 74. VID Code Voltages**

VID[4:0]	Voltage		VID[4:0]	Voltage	
00000	1.550 V		10000	1.150 V	
00001	1.525 V		10001	1.125 V	
00010	1.500 V		10010	1.100 V	
00011	1.475 V		10011	1.075 V	
00100	1.450 V		10100	1.050 V	
00101	1.425 V		10101	1.025 V	
00110	1.400 V		10110	1.000 V	
00111	1.375 V		10111	0.975 V	
01000	1.350 V		11000	0.950 V	
01001	1.325 V		11001	0.925 V	
01010	1.300 V		11010	0.900 V	
01011	1.275 V		11011	0.875 V	
01100	1.250 V		11100	0.850 V	
01101	1.225 V		11101	0.825 V	
01110	1.200 V		11110	0.800 V	
01111	1.175 V		11111	Off	

#### 9.6.2.1.5 Isochronous Relief Time

The Isochronous Relief Time (IRT) is the amount of time the processor driver must count after each FID change step in a given P-state transition. During each FID change step, system memory is inaccessible to bus masters. While the processor driver is counting IRT between FID change steps, bus masters have access to system memory. Table 75 lists the IRT values and their corresponding times.

**Table 75. IRT Values**

IRT (Bits 31–30)	Time
00	10 $\mu$ s
01	20 $\mu$ s
10	40 $\mu$ s

**Table 75. IRT Values**

IRT (Bits 31–30)	Time
11	80 $\mu$ s (BIOS default)

**9.6.2.2 \_PSS Status Field**

The status field of each performance state definition within the \_PSS object contains the information required by the processor driver to verify that the transition has completed based upon a read of the FIDVID\_STATUS MSR. The \_PSS Status field is 32 bits, and the BIOS must map the FIDVID\_STATUS MSR CurrVID and CurrFID information to the \_PSS status field as shown in Table 76 on page 284.

**Table 76. \_PSS Status Field**

Bit	Name	Description
5–0	CurrFID	The FID value associated with the OS-requested P-state
10–6	CurrVID	The VID value associated with the OS-requested P-state
31–11	Reserved	These bits must be 0 and are ignored by the processor driver.

The CurrFID and CurrVID fields of the FIDVID\_STATUS MSR are valid only when the FidVidPending bit is 0.

**9.6.3 \_PPC (Performance Present Capabilities)**

The \_PPC object is used to limit the maximum P-state that the operating system can use at any given time. This object follows the \_PSS object in the \\_PR name space for the operating systems that support the ACPI 2.0 processor P-state objects.

\_PPC is generically described in Section 8.3.3.3 of the ACPI 2.0 specification.

This section provides BIOS specifics regarding use of \_PPC.

If the OEM and BIOS vendor do not implement a \_PPC that limits the maximum P-state under certain conditions, then the BIOS must always return a value of 0 for the \_PPC object. When a value of 0 is returned for the \_PPC object, all P-states specified by the \_PSS object are available.

Operating systems with native support for processor P-state transitions and ACPI Thermal zones make use of reduced P-states for passive cooling before making use of “throttling” with the stop grant state. This is the case even when the \_PPC object returns a value of 0.

**9.6.3.1 Using \_PPC to Limit the Maximum Processor P-State When battery Powered**

Processor performance exceeds the demands of most typical notebook PC applications. However, there are basically idle scenarios where applications run instructions that incorrectly cause the

operating system to conclude that the processor is 100% utilized. This causes the operating system to force the processor to its maximum performance state, wasting power with no benefit to the user.

Also, static power consumption of the processor increases when the processor is operating in a P-state above the minimum voltage level supported by the processor.

Therefore, it will increase the system's battery-powered runtime to limit the maximum performance state available when battery-powered.

### 9.6.3.2 Implementation Overview

An ACPI-compliant General-Purpose Event (GPE) input of the chipset (typically the I/O Hub) is dedicated to causing an SCI whenever the notebook power source changes. This input will be referred to as AC available (ACAV). The system can control ACAV directly based upon the presence of an AC adapter, or the embedded controller (EC) in the system can control ACAV.

Whenever the AC adapter is inserted or removed from the system, the chipset logic associated with ACAV sets the status bit associated with ACAV, and if enabled causes an SCI to be asserted. Note: the control method associated with ACAV status could be implemented in the EC. A chipset GPE dedicated for AC adapter status is used for this description.

The BIOS provides the following ACPI ASL control methods and objects:

- `_PPC` (Performance Present Capabilities)
- `_PSR` (Power Source) is described in Section 11.3.1 of the ACPI 2.0 specification. `_PSR` returns:
  - 0x00000000 if the system is running on battery power.
  - 0x00000001 if the system is running from an AC adapter.
- `\_SB.AC` is a BIOS-defined device object for the AC adapter.
- `_L12` is the control method associated with GPE 12.

The flow associated with AC adapter insertion or removal is:

- AC adapter is inserted or removed
- The GPE Status bit associated with ACAV is asserted (for this example, GPE 12)
- An SCI (system control interrupt) is issued
- The OS/ACPI driver determines that GPE 12 was asserted and runs the associated control method (`_L12`)
- `_L12`
  - Issues a `Notify(\_PR.CPU0, 0x80)` which forces the OS to re-evaluate the `_PPC` object.
    - `_PPC` reads the state of the ACAV pin and returns:
      - 0 if the system is AC-powered. All P-states are available.

n if the system is battery-powered, where n is the highest P-state is supported when battery-powered. Only P-states n and lower are available. For example, if a processor supports 3 P-states:

P0 = 2400 MHz at 1.2 V

P1 = 1600 MHz at 1.1 V

P2 = 800 MHz at 1.0 V

Then n = 2 will limit the maximum P-state to 800 MHz at 1.0 V when the system is battery-powered.

- OS takes into account the available P-states and if necessary performs a P-state transition.
- Issues a Notify(\\_SB.AC, 0x80) where \\_SB.AC is the AC adapter device object.
  - OS runs \_PSR to determine if the current power source is the AC adapter. PSR reads the state of the ACAV pin and returns:
    - 1 if the system is AC-powered.
    - 0 if the system is battery-powered.
- Normal operation continues.

#### 9.6.4 PSTATE\_CNT

PSTATE\_CNT is an entry in the Fixed ACPI Description Table (FADT). A PSTATE\_CNT entry of 0 informs the operating system that the BIOS in System Management Mode (SMM) does not perform P-state transitions.

The BIOS must report a value of 0 in the PSTATE\_CNT field of the FADT. The only BIOS-controlled P-state transition, if any, must be performed near the beginning of the POST routine before control is passed to the operating system. All subsequent transitions are made by system software not the BIOS. System software is either the AMD PowerNow! technology software (for Microsoft® operating systems prior to the Windows® XP operating system) or the operating system and associated processor driver (for the Windows XP operating system and subsequent Microsoft operating systems).

**Note:** SMM is not used to perform P-state transitions.

#### 9.6.5 CST\_CNT

CST\_CNT is an entry in the FADT. If non-zero, this field contains the value the operating system writes to the SMI\_CMD register to indicate operating system support for the \_CST object and C States Changed notification.

The BIOS must report a value of 0 in the CST\_CNT field of the FADT. AMD platforms only support ACPI 1.0b processor power state functionality. Processor power states are referred to as “C” states by the ACPI specifications.

## 9.7 BIOS Support for AMD PowerNow!™ Software with Legacy Operating Systems

If BIOS determines it has a processor and system that support P-state transitions, it provides a PSB that is comprised of a header and the processor specific Performance State Table (PST) that matches the processor in the system. The PSB must be 16-byte aligned in memory.

If the BIOS cannot determine that the system and processor support P-state transitions as defined in 9.5.1 on page 266, then the BIOS must not provide a PSB. Changing the PSB Signature field to all 0s is an effective way of removing the PSB.

Table 77 defines the PSB and PST structures.

**Table 77. Performance State Block Structure**

Name	Length (Bytes)	Field Purpose
<b>PSB Header</b>		
Signature	10	Start of the PSB - Always set this field to "AMDK7PNOW!"
TableVersion	1	The version of the table as defined by AMD. Set to 14h for version 1.4. If AMD PowerNow! software does not recognize the TableVersion, it will not attempt to make P-state transitions.
Flags	1	All bits are reserved and written to zero.
VST	2	Voltage Stabilization Time. The amount of time required for the processor's core voltage regulator to increase the processor's core voltage the increment specified by MVS. VST is specified in 20 $\mu$ s increments. The default is 05h (100 $\mu$ s). All other values are reserved.
Reserved1	1	Bits 1–0: Ramp Voltage Offset (RVO) has the same definition as for the _PSS ACPI object. Bits 3–2: Isochronous Relief Time (IRT) has the same definition as for the _PSS ACPI object. Bits 5–4: Maximum Voltage Step (MVS) has the same definition as for the _PSS ACPI object. Bits 7–6: Number of available P-states when the system is battery powered has the following definition: 00b = all P-states are available 01b = only the minimum P-state is available 10b = 2 lowest P-states are available 11b = 3 lowest P-states are available.
NumPST	1	The total number of PSTs in the PSB. This value must be set to 01h.
<b>PST Header</b>		
CPUID	4	CPUID (EAX of CPUID extended function 8000_0001h) of the processor that this PST belongs to. This provides processor family, extended family, model, extended model, and stepping (revision) fields.

**Table 77. Performance State Block Structure (Continued)**

Name	Length (Bytes)	Field Purpose
PLL Lock Time	1	Processor PLL Lock time in microseconds. The PLL Lock time is specified in the processor data sheet addendum. For example, PLL_LOCK_TIME of 2 $\mu$ s is represented as 00000010b.
MaxFID	1	MaxFID[5:0] as reported by the MaxFID field of the FIDVID_STATUS MSR.
MaxVID	1	MaxVID[4:0] as reported by the MaxVID field of the FIDVID_STATUS MSR.
NumPStates	1	The number of FID, VID combinations in the PST. This field must be $\geq 1$ h.
FID	1	FID[5:0] code corresponding to the frequency of the lowest performance state that the processor supports. This is defined in the processor data sheet.
VID	1	VID[4:0] code corresponding to the voltage of the lowest performance state that the processor supports. This is defined in the processor data sheet.
-----	-----	FID, VID pairs are concatenated here so that the total number of pairs is equal to the NumPstates field. Each subsequent FID, VID pair is appended in ascending order (i.e., the last FID,VID pair corresponds to the maximum performances state supported by the processor).

For processors with a family number of less than Fh, refer to BIOS Requirements for AMD PowerNow! Technology Application Note, order# 25264.

## 9.8 System Configuration for Power Management

### 9.8.1 Chipset Configuration for Power Management

Chipset configuration is covered in chipset related documentation. The BIOS is required to:

- Ensure that the SMAF code to STPCLK and Stop Grant mapping for all components based on HyperTransport technology is configured as described in Table 68 on page 263.
- For systems with 939 pin or 754 pin processors, the HyperTransport links are tristated in response to LDTSTOP\_L assertion.
- Chipset power management features are appropriately enabled based upon system class.
- LDTSTOP\_L assertion for FID changes during P-state transitions:
  - For Revision B processors is at least 4 $\mu$ s but less than 10  $\mu$ s for single processor 940 pin processor systems.
  - For Revision C and later revisions is at least 2 $\mu$ s. Note: for UMA chipsets 2 $\mu$ s is ideal and may be required in some cases. Note: Links need to be 8-bits wide or larger and 400 MHz or greater.
  - Must be less than 6 $\mu$ s for chipsets that support isochronous streams (e.g. UMA graphics).



- LDTSTOP\_L assertion for link width/frequency changes during initialization is a minimum of 2  $\mu$ s for all processors. Note: For Links operating at 200 MHz no DMA traffic can be in progress.
- LDTSTOP\_L assertion during FID\_Change is 10  $\mu$ s for 940 pin processor systems.
- LDTSTOP\_L assertion during C3 is a minimum of 1  $\mu$ s for all processors.

## 9.8.2 Processor Configuration for Power Management

BIOS configures the processor to enable power management during the POST routine. Table 78 on page 289 lists processor memory system configuration register settings that the BIOS must initialize.

**Table 78. Configuration Register Settings for Power Management**

Functionality	Register	Mobile Setting	UP Desktop Setting	Multiprocessor Setting
Do not enable HyperTransport™ links to be tristated in response to LDTSTOP_L assertion except for revision E and later mobile systems.	LDT0 Link Control (function 0: offset 84h[13]) (LdtStopTriEn) LDT1 Link Control (function 0: offset A4h[13]) (LdtStopTriEn) LDT2 Link Control (function 0: offset C4h[13]) (LdtStopTriEn)	Revision D and earlier revisions 0b  Revision E and later 1b	0b	0b
Do not enable overriding tristating of HyperTransport™ clocks in response to LDTSTOP_L assertion except for Revision E and later mobile systems with UMA Graphics.	LDT0 Link Control (function 0: offset 84h[15]) (LdtStopTriClkOvr) LDT1 Link Control (function 0: offset A4h[15]) (LdtStopTriClkOvr) LDT2 Link Control (function 0: offset C4h[15]) (LdtStopTriClkOvr)	Revision D and earlier revisions or discrete graphics 0b  Revision E and later UMA graphics 1b	0b	0b
Do not enable HyperTransport™ Receivers to be tristated in response to LDTSTOP_L assertion except for Revision E and later mobile systems.	LDT0 Link Control (function 0: offset 84h[11]) (LdtStopRcvDis) LDT1 Link Control (function 0: offset A4h[11]) (LdtStopRcvDis) LDT2 Link Control (function 0: offset C4h[11]) (LdtStopRcvDis)	Revision D and earlier revisions 0b  Revision E and later revisions 1b	0b	0b

**Table 78. Configuration Register Settings for Power Management (Continued)**

Functionality	Register	Mobile Setting	UP Desktop Setting	Multiprocessor Setting
Map STPCLK actions to SMAF codes: SMAF 0 = C2 SMAF 1 = C3 SMAF 2 = VID/FID Change SMAF 3 = S1	Power Management Control Low (function 3: offset 80h)	Revision D and earlier revisions discrete graphics 3307_3331h  Revision E and later discrete graphics 6B07_6B31h  UMA graphics 3307_3331	2307_0000h	Revision B: 2117_0000h  Revision C: VID/FID capable <sup>2</sup> 2307_0000h Not VID/FID capable <sup>2</sup> 2303_0000h
Map STPCLK actions to SMAF codes: SMAF 4 = S3 SMAF 5 = Throttling SMAF 6 = S4/S5 SMAF 7 is not used for STPCLK, the corresponding field is used for HALT.	Power Management Control High (function 3: offset 84h)	Discrete graphics 6113_3113h  UMA graphics 3113_3113h	2113_2113h	Revision B: 0013_0013h Revision C <sup>1</sup> : 0013_2113h
BCLK and LCLK PLL frequency lock	Clock Power/Timing Low (function 3: offset D4h)	Revision B: 000D_0001h Revisions C and CG with unbuffered memory: 000D_0701h Revisions C CG and D with registered memory: 04E2_0707h Revision E 000D_A701h		Revision B: 000D_0001h Revisions C, CG and D: 04E2_0707h Revision E 04E2_A707h
Sets "Ramp VID" (0 mV) and voltage regulator stabilization time (0 $\mu$ s). The processor driver performs this function, not hardware. <sup>3</sup>	Clock Power/Timing High (function 3: offset D8h [30:28], [19:0]) (RampVIDOf) (VSTime)	000b[30:28] 0_0000h[19:0]	000b[30:28] 0_0000h[19:0]	000b[30:28] 0_0000h[19:0]
Sets the AltVID value.	Clock Power/Timing High (function 3: offset D8h [24:20]) (AltVid)	See the "Power and Thermal" data sheet	N/A	N/A

**Table 78. Configuration Register Settings for Power Management (Continued)**

Functionality	Register	Mobile Setting	UP Desktop Setting	Multiprocessor Setting
Do not enable tristating of memory clock except for revision E and later systems with discrete graphics.	DRAM PDL Low (Function 2: offset 98h[27])	Revision D and earlier revisions or UMA graphics 0b  Revision E and later discrete graphics 1b	0b	0b
Enable disabling of the DRAM compensation circuitry for revision E and later revisions	DRAM PDL Low (Function 2: offset 98h[28])	Revision D and earlier revisions 0b  Revision E and later revisions 1b	0b	0b
Enable DRAM power down mode.	DRAM Configuration Low (Function 2: offset 90h[31:30])	Based on motherboard DRAM chip select routing see "DRAM Configuration High Register Function 2: Offset 94h"	Based on motherboard DRAM chip select routing see "DRAM Configuration High Register Function 2: Offset 94h"	Based on motherboard DRAM chip select routing see "DRAM Configuration High Register Function 2: Offset 94h"
Enable tristating of the DRAM address and command when DIMMs are in the precharge power down state	DRAM Configuration Low (Function 2: offset 90h[7])	Revision D and earlier revisions 0 Revision E and later revisions 1b	Revision D and earlier revisions 0 Revision E and later revisions 1b	0b

**Table 78. Configuration Register Settings for Power Management (Continued)**

Functionality	Register	Mobile Setting	UP Desktop Setting	Multiprocessor Setting
<b>Notes:</b> <ol style="list-style-type: none"> <li>1. Optionally, a value of 2113_2113h can be used. This will significantly reduce processor power during halt and may reduce system performance. To ensure that system performance loss is kept to a minimum in this state, ClkRampHys (Function 3, Offset D4h) should be programmed to 2 us (0111b.)</li> <li>2. VID/FID capability is determined by executing the CPUID extended function 8000_0007h. If EDX[2:1]=11b, the processor is capable of changing VID/FID. If EDX[2:1]=00b, the processor is not capable of changing VID/FID.</li> <li>3. For revision E and later mobile systems that enable AltVID VSTIME should be programmed to account for the time it takes to ramp the voltage from AltVID to the VID for the 800MHz P-State.  <math display="block">VSTIME = ((800 \text{ MHz VID}) - \text{AltVID}) * 1\mu\text{s/mV}</math> </li> </ol>				

### 9.8.2.1 Additional BIOS Support Requirements for S3

During the S3 state, power is removed from the processor's CPU core, host bridge, and memory controller. It is required that the BIOS restore the state of the processor's memory controller upon resume from S3 prior to having the memory controller bring system memory out of self-refresh mode. The system must provide non volatile storage for the memory controller configuration registers during the S3 state. For UP and DP systems, this storage will be provided in the chipset. There are two opportunities for the BIOS to store the memory controller configuration to this nonvolatile memory.

- During POST as the memory controller is being initially configured—this is preferred, since it does not require SMM support.
- In SMM just prior to entry to the S3 state—BIOS uses an SMI trap on writes to the ACPI PM1 control register to get control of the system just prior to entry to S3. After storing the memory controller configuration registers, I/O instruction restart is used to place the system into the S3 state. This is not preferred because it relies on SMM.

Processor Function 2 configuration space registers at offsets 40–98h must be stored in nonvolatile RAM for support of resume from S3.

### 9.8.2.2 ACPI Tables/Objects

FADT entries for C-state and throttling support and throttling is listed in Table 79 on page 292. Some legacy operating systems will not use C3 if support for C2 is not declared.

**Table 79. FADT Table Entries**

Field	Mobile	UP Desktop	Multiprocessor
PSTATE_CNT	0	0	0
CST_CNT	0	0	0

**Table 79. FADT Table Entries**

Field	Mobile	UP Desktop	Multiprocessor
P_LVL2_LAT	> 100 if C2 is not supported 1 if C2 is supported	> 100	> 100
P_LVL3_LAT	> 1000 if C3 is not supported 10 if C3 is supported	> 1000	> 1000
DUTY_WIDTH	0 if _PSV is not used 3 if _PSV is used	0 if _PSV is not used 3 if _PSV is used	0



## 10 Performance Monitoring

---

AMD Athlon™ 64 and AMD Opteron™ Processors support the performance-monitoring features introduced in earlier processor implementations. These features allow the selection of events to be monitored, and include a set of corresponding counter registers that track the frequency of monitored events. Software tools can use these features to identify performance bottlenecks, such as sections of code that have high cache-miss rates or frequently mis-predicted branches. This information can then be used as a guide for improving or eliminating performance problems through software optimizations or hardware-design improvements.

For a general description of how to use these performance monitoring features, refer to the “Debug and Performance Resources” section in Volume 2: System Programming of the *AMD64 Architecture Programmers Manual*, order# 24593.

### 10.1 Performance Counters

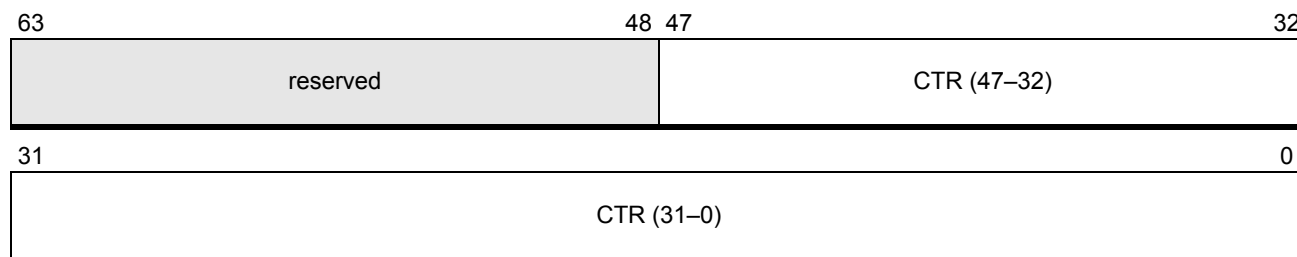
The processor provides four 48-bit performance counters. Each counter can monitor a different event. The accuracy of the counters is not ensured. Some events are reserved. When a reserved event is selected, the results are undefined.

Performance counters are used to count specific processor events, such as data-cache misses, or the duration of events, such as the number of clocks it takes to return data from memory after a cache miss. During event counting, the processor increments the counter when it detects an occurrence of the event. During duration measurement, the processor counts the number of processor clocks it takes to complete an event. Each performance counter can be used to count one event, or measure the duration of one event at a time.

In addition to RDMSR instruction, the PerfCtrn registers can be read using a special *read performance-monitoring counter* instruction, RDPMC. The RDPMC instruction loads the contents of the PerfCtrn register specified by the ECX register, into the EDX register and the EAX register.

Writing the performance counters can be useful if software wants to count a specific number of events, and then trigger an interrupt when that count is reached. An interrupt can be triggered when a performance counter overflows. Software should use the WRMSR instruction to load the count as a two's-complement negative number into the performance counter. This causes the counter to overflow after counting the appropriate number of times.

The performance counters are not assured of producing identical measurements each time they are used to measure a particular instruction sequence, and they should not be used to take measurements of very small instruction sequences. The RDPMC instruction is not serializing, and it can be executed out-of-order with respect to other instructions around it. Even when bound by serializing instructions, the system environment at the time the instruction is executed can cause events to be counted before the counter value is loaded into EDX:EAX.

**PerfCtr0–PerfCtr3 Registers C001\_0004h, C001\_0005h, C001\_0006h, C001\_0007h**

Bit	Name	Function	R/W
63–48	Reserved	RAZ	R
47–0	CTR	Counter value	R

## 10.2 Performance Event-Select Registers

Performance Event-Select registers (PerfEvtSel*i*) are used to specify the events counted by the performance counters, and to control other aspects of their operation. Each performance counter supported by the implementation has a corresponding event-select register that controls its operation. This section shows the format of the PerfEvtSel*n* registers.

To accurately start counting with the write that enables the counter, disable the counter when changing the event and then enable the counter with a second MSR write.

The edge count mode will increment the counter when a transition happens on the monitored event. If the event selected is changed without disabling the counter, an extra edge will be falsely detected when the first event is a static 0 and the second event is a static one. To avoid this false edge detection, disable the counter when changing the event and then enable the counter with a second MSR write.

The EVENT\_SELECT field specifies the event or event duration in a processor unit to be counted by the corresponding PerfCtr*i* register. The UNIT\_MASK field is used to further specify or qualify a monitored event. The events that can be counted are implementation dependent. For more up-to-date information on the events in the latest processor revisions, refer to the *Revision Guide for the AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25759.

The performance counter registers can be used to track events in the Northbridge. When a Northbridge event is selected using one of the Performance Event-Select registers in either core of a dual core processor, that Performance Event-Select register and the corresponding Performance Counter register cannot be used by the other core. Monitoring of Northbridge events should only be performed by one core in a dual core processor.

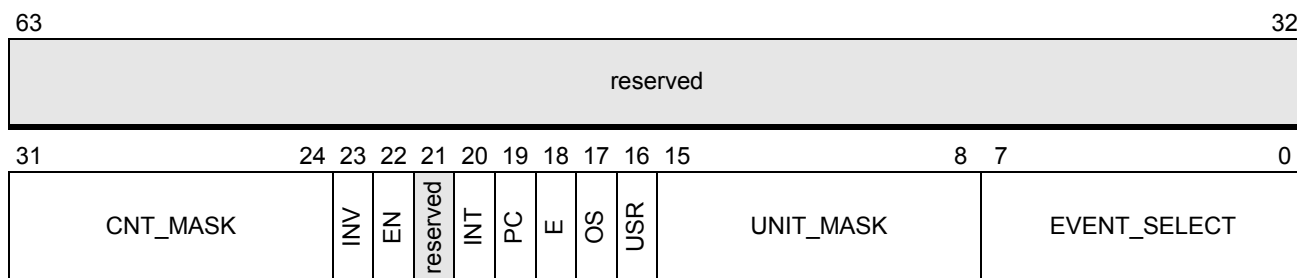
Care must also be taken when measuring Northbridge or other non-processor-specific events under conditions where the processor may go into Halt mode during the measurement period. For instance, one may wish to monitor DRAM traffic due to DMA activity from a disk or graphics adaptor. This entails running some event counter monitoring code on the processor, where such code accesses the



counters at the beginning and end of the measurement period, or may even sample them periodically throughout the measurement period. Such code typically gives up the processor during each measurement interval. If there is nothing else for the OS to run on that particular processor at that time, it may halt the processor until it is needed. Under these circumstances, the clock for the counter logic will be stopped, hence the counters will not count the events of interest. To prevent this, simply run a low-priority background process that will keep the processor busy during the period of interest.

## PerfEvtSel0–PerfEvtSel3 Registers

**C001\_0000h, C001\_0001h,  
C001\_0002h, C001\_0003h**



Bit	Name	Function	R/W
63–32	reserved	RAZ	
31–24	CNT_MASK	Counter Mask	R/W
23	INV	Invert Counter Mask	R/W
22	EN	Enable Counter	R/W
21	reserved	SBZ	
20	INT	Enable APIC Interrupt	R/W
19	PC	Pin Control	R/W
18	E	Edge Detect	R/W
17	OS	Operating-System Mode	R/W
16	USR	User Mode	R/W
15–8	UNIT_MASK	Event Qualification	R/W
7–0	EVENT_SELECT	Unit and Event Selection	R/W

## Field Descriptions

**Event Selection (EVENT\_SELECT)**—Bits 7–0. The EVENT\_SELECT field specifies the event or event duration in a processor unit to be counted by the corresponding PerfCtr<sub>i</sub> register.

**Event Qualification (UNIT\_MASK)**—Bits 15–8. Each UNIT\_MASK bit further specifies or qualifies the event specified by EVENT\_MASK. All events selected by UNIT\_MASK are simultaneously monitored. Unless otherwise stated, the UNIT\_MASK values shown may be combined (logically ORed) to select any desired combination of the sub-events for a given event. In some cases, certain combinations can result in misleading counts, or the UNIT\_MASK value is an ordinal rather than a bit mask. These situations are described where

applicable, or should be obvious from the event descriptions. For events where no UNIT\_MASK table is shown, the UNIT\_MASK is not applicable and may be set to zeros.

**User Mode (USR)**—Bit 16. Count events in user mode (at CPL > 0).

**Operating-System Mode (OS)**—Bit 17. Count events in operating-system mode (at CPL = 0).

**Edge Detect (E)**—Bit 18. 1=Edge detection. 0=Level detection.

**Pin Control (PC)**—Bit 19. With a value of 1, toggles the performance monitor pin PM<sub>i</sub> when PerfCtr<sub>i</sub> register overflows. With a value of 0, toggles the performance monitor pin PM<sub>i</sub> each time PerfCtr<sub>i</sub> register is incremented.

**Enable APIC Interrupt (INT)**—Bit 20. Enables APIC interrupt when PerfCtr<sub>i</sub> register overflows.

**Enable Counter (EN)**—Bit 22. Enables performance monitor counter.

**Invert Counter Mask (INV)**—Bit 23. See CNT\_MASK.

**Counter Mask (CNT\_MASK)**—Bits 31–24. a) When CNT\_MASK is 0, the corresponding PerfCtr<sub>i</sub> register is incremented by the number of events occurring in a clock cycle. Maximum number of events in one cycle is 3. b) When CNT\_MASK values are from 1 to 3 and INV is 0, the corresponding PerfCtr<sub>i</sub> register is incremented by 1, if the number of events occurring in a clock cycle is greater than or equal to CNT\_MASK. When CNT\_MASK values are from 1 to 3 and INV is 1, the corresponding PerfCtr<sub>i</sub> register is incremented by 1, if the number of events occurring in a clock cycle is less than CNT\_MASK. c) CNT\_MASK values from 4 to 255 are reserved.

## 10.2.1 Performance Monitor Events

**Note:** *Speculative vs. Retired:* Several events may include speculative activity, meaning they may be associated with false-path instructions that are ultimately discarded due to a branch misprediction. Events associated with Retire reflect actual program execution. For events where the distinction may matter, these are explicitly labeled as one or the other.

### 10.2.1.1 Floating Point Unit Events

#### Event Select: 00h - Dispatched FPU Operations

The number of operations (uops) dispatched to the FPU execution pipelines. This event reflects how busy the FPU pipelines are. This includes all operations done by x87, MMX and SSE instructions, including moves. Each increment represents a one-cycle dispatch event; packed 128-bit SSE operations count as two ops; scalar operations count as one. Speculative. (See also event CBh).

Note: Since this event includes non-numeric operations it is not suitable for measuring MFLOPs.

UNIT_MASK	
01h	Add pipe ops

UNIT_MASK	
02h	Multiply pipe ops
04h	Store pipe ops
08h	Add pipe load ops
10h	Multiply pipe load ops
20h	Store pipe load ops

**Event Select: 01h - Cycles with no FPU Ops Retired**

The number of cycles in which no FPU operations were retired. Invert this (set the Invert control bit in the event select MSR) to count cycles in which at least one FPU operation was retired.

**Event Select: 02h - Dispatched Fast Flag FPU Operations**

The number of FPU operations that use the fast flag interface (e.g. FCOMI, COMISS, COMISD, UCOMISS, UCOMISD). Speculative.

**10.2.1.2 Load/Store Unit and TLB Events****Event Select: 20h - Segment Register Loads**

The number of segment register loads performed.

UNIT_MASK	
01h	ES
02h	CS
04h	SS
08h	DS
10h	FS
20h	GS
40h	HS

**Event Select: 21h - Pipeline Restart Due to Self-Modifying Code**

The number of pipeline restarts that were caused by self-modifying code (a store that hits any instruction that's been fetched for execution beyond the instruction doing the store).

**Event Select: 22h - Pipeline Restart Due to Probe Hit**

The number of pipeline restarts caused by an invalidating probe hitting on a speculative out-of-order load.

**Event Select: 23h - LS Buffer 2 Full**

The number of cycles that the LS2 buffer is full. This buffer holds stores waiting to retire as well as requests that missed the data cache and are waiting on a refill. This condition will stall further data cache accesses, although such stalls may be overlapped to some extent by independent instruction execution.

**Event Select: 24h - Locked Operations**

This event covers locked operations performed and their execution time. The execution time represented by the cycle counts is typically overlapped to a large extent with other instructions. The *non-speculative cycles* event is suitable for event-based profiling of lock operations that tend to miss in the cache.

UNIT_MASK	
01h	The number of locked instructions executed
02h	The number of cycles spent in speculative phase
04h	The number of cycles spent in non-speculative phase (including cache miss penalty)

**Event Select: 65h - Memory Requests by Type**

These events reflect accesses to uncacheable (UC) or write-combining (WC) memory regions (as defined by MTRR or PAT settings) and Streaming Store activity to WB memory. Both the WC and Streaming Store events reflect Write Combining buffer flushes, not individual store instructions. WC buffer flushes which typically consist of one 64-byte write to the system for each flush (assuming software typically fills a buffer before it gets flushed). A partially-filled buffer will require two or more smaller writes to the system. The WC event reflects flushes of WC buffers that were filled by stores to WC memory or streaming stores to WB memory. The Streaming Store event reflects only flushes due to streaming stores (which are typically only to WB memory). The difference between counts of these two events reflects the true amount of write events to WC memory.

UNIT_MASK	
01h	Requests to non-cacheable (UC) memory
02h	Requests to write-combining (WC) memory or WC buffer flushes to WB memory
80h	Streaming store (SS) requests

**10.2.1.3 Data Cache Events****Event Select: 40h - Data Cache Accesses**

The number of accesses to the data cache for load and store references. This may include certain microcode scratchpad accesses, although these are generally rare. Each increment represents an eight-byte access, although the instruction may only be accessing a portion of that. Speculative.

**Event Select: 41h - Data Cache Misses**

The number of data cache references which missed in the data cache. Speculative.

Except in the case of streaming stores, only the first miss for a given line is included - access attempts by other instructions while the refill is still pending are not included in this event. So in the absence of streaming stores, each event reflects one 64-byte cache line refill, and counts of this event are the same as, or very close to, the combined count for event 42h.

Streaming stores however will cause this event for every such store, since the target memory is not refilled into the cache. Hence this event should not be used as an indication of data cache refill activity - event 42h should be used for such measurements. (See event 65h for an indication of streaming store activity.) A large difference between events 41h (with all UNIT\_MASK bits set) and 42h would be due mainly to streaming store activity.

**Event Select: 42h - Data Cache Refills from L2 or System**

The number of data cache refills satisfied from the L2 cache (and/or the system), per the UNIT\_MASK. UNIT\_MASK bits 4:1 allow a breakdown of refills from the L2 by coherency state. UNIT\_MASK bit 0 reflects refills which missed in the L2, and provides the same measure as the combined sub-events of event 43h. Each increment reflects a 64-byte transfer. Speculative.

UNIT_MASK	
01h	Refill from System
02h	Shared-state line from L2
04h	Exclusive-state line from L2
08h	Owned-state line from L2
10h	Modified-state line from L2

**Event Select: 43h - Data Cache Refills from System**

The number of L1 cache refills satisfied from the system (system memory or another cache), as opposed to the L2. The UNIT\_MASK selects lines in one or more specific coherency states. Each increment reflects a 64-byte transfer. Speculative.

UNIT_MASK	
01h	Invalid
02h	Shared
04h	Exclusive
08h	Owned
10h	Modified

**Event Select: 44h - Data Cache Lines Evicted**

The number of L1 data cache lines written to the L2 cache or system memory, having been displaced by L1 refills. The UNIT\_MASK may be used to count only victims in specific coherency states. Each increment represents a 64-byte transfer. Speculative.

In most cases, L1 victims are moved to the L2 cache, displacing an older cache line there. Lines brought into the data cache by PrefetchNTA instructions, however, are evicted directly to system memory (if dirty) or invalidated (if clean). There is no provision for measuring this component by itself. The Invalid case (UNIT\_MASK value 01h) reflects the replacement of lines that would have been invalidated by probes for write operations from another processor or DMA activity.

UNIT_MASK	
01h	Invalid
02h	Shared
04h	Exclusive
08h	Owned
10h	Modified

**Event Select: 45h - L1 DTLB Miss and L2 DLTB Hit**

The number of data cache accesses that miss in the L1 DTLB and hit in the L2 DTLB. Speculative.

**Event Select: 46h - L1 DTLB and L2 DLTB Miss**

The number of data cache accesses that miss in both the L1 and L2 DTLBs. Speculative.

**Event Select: 47h - Misaligned Accesses**

The number of data cache accesses that are misaligned. These are accesses which cross an eight-byte boundary. They incur an extra cache access (reflected in event 40h), and an extra cycle of latency on reads. Speculative.

**Event Select: 48h - Microarchitectural Late Cancel of an Access****Event Select: 49h - Microarchitectural Early Cancel of an Access****Event Select: 4Ah - Single-bit ECC Errors Recorded by Scrubber**

The number of single-bit errors corrected by either of the error detection/correction mechanisms in the data cache.

UNIT_MASK	
01h	Scrubber error
02h	Piggyback scrubber errors

**Event Select: 4Bh - Prefetch Instructions Dispatched**

The number of prefetch instructions dispatched by the decoder. Speculative. Such instructions may or may not cause a cache line transfer. Any Dcache and L2 accesses, hits and misses by prefetch instructions are included in those types of events.

UNIT_MASK	
01h	Load (Prefetch, PrefetchT0/T1/T2)
02h	Store (PrefetchW)
04h	NTA (PrefetchNTA)

**Event Select: 4Ch - DCACHE Misses by Locked Instructions**

The number of data cache misses incurred by locked instructions. (The total number of locked instructions may be obtained from event 24h.)

Such misses may be satisfied from the L2 or system memory, but there is no provision for distinguishing between the two. When used for event-based profiling, this event tends to occur very close to the offending instructions. (See also event 24h.) This event is also included in the basic Dcache miss event (41h).

UNIT_MASK	
02h	Data cache misses by locked instructions

**10.2.1.4 L2 Cache and System Interface Events****Event Select: 67h - Data Prefetcher**

These events reflect requests made by the data prefetcher. UNIT\_MASK bit 1 counts total prefetch requests, while bit 0 counts requests where the target block is found in the L2 or data cache. The difference between the two represents actual data read (in units of 64-byte cache lines) from the system by the prefetcher. This is also included in the count of event 7Fh, UNIT\_MASK bit 0 (combined with other L2 fill events).

UNIT_MASK	
01h	Cancelled prefetches
02h	Prefetch attempts

**Event Select: 6Ch - System Read Responses by Coherency State**

The number of responses from the system for cache refill requests. The UNIT\_MASK may be used to select specific cache coherency states. Each increment represents one 64-byte cache line transferred from the system (DRAM or another cache, including another core on the same node) to the data cache, instruction cache or L2 cache (for data prefetcher and TLB table walks). Modified-state

responses may be for Dcache store miss refills, PrefetchW software prefetches, hardware prefetches for a store-miss stream, or Change-to-Dirty requests that get a dirty (Owned) probe hit in another cache. Exclusive responses may be for any Icache refill, Dcache load miss refill, other software prefetches, hardware prefetches for a load-miss stream, or TLB table walks that miss in the L2 cache. Shared responses may be for any of those that hit a clean line in another cache.

UNIT_MASK	
01h	Exclusive
02h	Modified
04h	Shared

### Event Select: 6Dh - Quadwords Written to System

The number of quadword (8-byte) data transfers from the processor to the system. These may be part of a 64-byte cache line writeback or a 64-byte dirty probe hit response, each of which would cause eight increments, or a partial or complete Write Combining buffer flush (Sized Write), which could cause from one to eight increments.

UNIT_MASK	
01h	Quadword write transfer

### Event Select: 7Dh - Requests to L2 Cache

The number of requests to the L2 cache for Icache or Dcache fills, or page table lookups for the TLB. These events reflect only read requests to the L2. Writes to the L2 are indicated by event 7Fh. These include some amount of retries associated with address or resource conflicts. Such retries tend to occur more as the L2 gets busier, and in certain extreme cases (such as large block moves that overflow the L2) these extra requests can dominate the event count.

These extra requests are not a direct indication of performance impact - they simply reflect opportunistic accesses that don't complete. But because of this, they are not a good indication of actual cache line movement. The Icache and Dcache miss and refill events (81h, 82h, 83h, 41h, 42h, 43h) provide a more accurate indication of this, and are the preferred way to measure such traffic.

UNIT_MASK	
01h	IC fill
02h	DC fill
04h	TLB fill (page table walks)
08h	Tag snoop request
10h	Cancelled request



**Event Select: 7Eh - L2 Cache Misses**

The number of requests that miss in the L2 cache. This may include some amount of speculative activity, as well as some amount of retried requests as described in event 7Dh. The IC-fill-miss and DC-fill-miss events tend to mirror the Icache and Dcache refill-from-system events (83h and 43h, respectively), and tend to include more speculative activity than those events.

UNIT_MASK	
01h	IC fill
02h	DC fill (includes possible replays, whereas event 41h does not)
04h	TLB page table walk

**Event Select: 7Fh - L2 Fill/Writeback**

The number of lines written into the L2 cache due to victim writebacks from the Icache or Dcache, TLB page table walks and the hardware data prefetcher (UNIT\_MASK bit 0); or writebacks of dirty lines from the L2 to the system (UNIT\_MASK bit 1). Each increment represents a 64-byte cache line transfer.

**Note:** *Victim writebacks from the Dcache may be measured separately using event 44h. However, this is not quite the same as the Dcache component of event 7Fh, the main difference being PrefetchNTA lines. When these are evicted from the Dcache due to replacement, they are written out to system memory (if dirty) or simply invalidated (if clean), rather than being moved to the L2 cache.*

UNIT_MASK	
01h	L2 fills (victims from L1 caches, TLB page table walks and data prefetches)

**10.2.1.5 Instruction Cache Events**

All instruction cache events are speculative.

**Event Select: 80h - Instruction Cache Fetches**

The number of instruction cache accesses by the instruction fetcher. Each access is an aligned 16 byte read, from which a varying number of instructions may be decoded.

**Event Select: 81h - Instruction Cache Misses**

The number of instruction fetches that miss in the instruction cache. This is typically equal to or very close to the sum of events 82h and 83h. Each miss results in a 64-byte cache line refill.

**Event Select: 82h - Instruction Cache Refills from L2**

The number of instruction cache refills satisfied from the L2 cache. Each increment represents one 64-byte cache line transfer.

**Event Select: 83h - Instruction Cache Refills from System**

The number of instruction cache refills from system memory (or another cache). Each increment represents one 64-byte cache line transfer.

**Event Select: 84h - L1 ITLB Miss, L2 ITLB Hit**

The number of instruction fetches that miss in the L1 ITLB but hit in the L2 ITLB.

**Event Select: 85h - L1 ITLB Miss, L2 ITLB Miss**

The number of instruction fetches that miss in both the L1 and L2 ITLBs.

**Event Select: 86h - Pipeline Restart Due to Instruction Stream Probe**

The number of pipeline restarts caused by invalidating probes that hit on the instruction stream currently being executed. This would happen if the active instruction stream was being modified by another processor in an MP system - typically a highly unlikely event.

**Event Select: 87h - Instruction Fetch Stall**

The number of cycles the instruction fetcher is stalled. This may be for a variety of reasons such as branch predictor updates, unconditional branch bubbles, far jumps and cache misses, among others. May be overlapped by instruction dispatch stalls or instruction execution, such that these stalls don't necessarily impact performance.

**Event Select: 88h - Return Stack Hits**

The number of near return instructions (RET or RET Iw) that get their return address from the return address stack (i.e. where the stack has not gone empty). This may include cases where the address is incorrect (return mispredicts). This may also include speculatively executed false-path returns. Return mispredicts are typically caused by the return address stack underflowing. However, they may also be caused by an imbalance in calls vs. returns, such as doing a call but then popping the return address off the stack.

**Note:** *This event cannot be reliably compared with events C9h and CAh (such as to calculate percentage of return mispredicts due to an empty return address stack), since it may include speculatively executed false-path returns that are not included in those retire-time events.*

**Event Select: 89h - Return Stack Overflows**

The number of (near) call instructions that cause the return address stack to overflow. When this happens, the oldest entry is discarded. This count may include speculatively executed calls.

### 10.2.1.6 Execution Unit Events

#### Event Select: 26h - Retired CLFLUSH Instructions

The number of CLFLUSH instructions retired.

#### Event Select: 27h - Retired CPUID Instructions.

The number of CPUID instructions retired.

#### Event Select: 76h - CPU Clocks not Halted

The number of clocks that the CPU is not in a halted state (due to STPCLK or a HALT instruction).

***Note:** This event allows system idle time to be automatically factored out from IPC (or CPI) measurements, providing the OS halts the CPU when going idle. If the OS goes into an idle loop rather than halting, such calculations will be influenced by the IPC of the idle loop.*

#### Event Select: C0h - Retired Instructions

The number of instructions retired (execution completed and architectural state updated). This count includes exceptions and interrupts – each exception or interrupt is counted as one instruction.

#### Event Select: C1h - Retired uops

The number of micro-ops retired. This includes all processor activity (instructions, exceptions, interrupts, microcode assists, etc.).

#### Event Select: C2h - Retired Branch Instructions

The number of branch instructions retired. This includes all types of architectural control flow changes, including exceptions and interrupts.

#### Event Select: C3h - Retired Mispredicted Branch Instructions

The number of branch instructions retired, of any type, that were not correctly predicted. This includes those for which prediction is not attempted (far control transfers, exceptions and interrupts).

#### Event Select: C4h - Retired Taken Branch Instructions

The number of taken branches that were retired. This includes all types of architectural control flow changes, including exceptions and interrupts.

#### Event Select: C5h - Retired Taken Branch Instructions Mispredicted

The number of retired taken branch instructions that were mispredicted.

**Event Select: C6h - Retired Far Control Transfers**

The number of far control transfers retired including far call/jump/return, IRET, SYSCALL and SYSRET, plus exceptions and interrupts. Far control transfers are not subject to branch prediction.

**Event Select: C7h - Retired Branch Resyncs**

The number of resync branches. These reflect pipeline restarts due to certain microcode assists and events such as writes to the active instruction stream, among other things. Each occurrence reflects a restart penalty similar to a branch mispredict. Relatively rare.

**Event Select: C8h - Retired Near Returns**

The number of near return instructions (RET or RET Iw) retired.

**Event Select: C9h - Retired Near Returns Mispredicted**

The number of near returns retired that were not correctly predicted by the return address predictor. Each such mispredict incurs the same penalty as a mispredicted conditional branch instruction.

**Event Select: CAh - Retired Indirect Branches Mispredicted**

The number of indirect branch instructions retired where the target address was not correctly predicted.

**Event Select: CBh - Retired MMX/FP Instructions**

The number of MMX, SSE or X87 instructions retired. The UNIT\_MASK allows the selection of the individual classes of instructions as given in the table. Each increment represents one complete instruction.

**Note:** Since this event includes non-numeric instructions, it is not suitable for measuring MFLOPS.

UNIT_MASK	
01h	x87 instructions
02h	MMX™ and 3DNow!™ instructions
04h	Packed SSE and SSE2 instructions
08h	Scalar SSE and SSE2 instructions

**Event Select: CCh - Retired Fastpath Double op Instructions**

UNIT_MASK	
01h	With low op in position 0
02h	With low op in position 1
04h	With low op in position 2

**Event Select: CDh - Interrupts-Masked Cycles**

The number of processor cycles where interrupts are masked (EFLAGS.IF = 0). Using edge-counting with this event gives the number of times IF is cleared. Dividing the cycle-count value by this value gives the average length of time that interrupts are disabled on each instance. Compare the edge count with event CFh to determine how often interrupts are disabled for interrupt handling vs. other reasons (e.g., critical sections).

**Event Select: CEh - Interrupts-Masked Cycles with Interrupt Pending**

The number of processor cycles where interrupts are masked (EFLAGS.IF = 0) and an interrupt is pending. Using edge-counting with this event and comparing the resulting count with the edge count for event CDh gives the proportion of interrupts for which handling is delayed due to prior interrupts being serviced, critical sections, etc. The cycle count value gives the total amount of time for such delays. The cycle count divided by the edge count gives the average length of each such delay.

**Event Select: CFh - Interrupts Taken**

The number of hardware interrupts taken. This does not include software interrupts (INT n instruction).

**Event Select: D0h - Decoder Empty**

The number of processor cycles where the decoder has nothing to dispatch (typically waiting on an instruction fetch that missed the Icache or for the target fetch after a branch mispredict).

**Event Select: D1h - Dispatch Stalls**

The number of processor cycles where the decoder is stalled for any reason (has one or more instructions ready but can't dispatch them due to resource limitations in execution). This is the combined effect of events D2h - DAh, some of which may overlap. This event reflects the net stall cycles. The more common stall conditions (events D5h, D6h, D7h, D8h, and to a lesser extent D2) may overlap considerably. The occurrence of these stalls is highly dependent on the nature of the code being executed (instruction mix, memory reference patterns, etc.).

**Event Select: D2h - Dispatch Stall for Branch Abort to Retire**

The number of processor cycles the decoder is stalled waiting for the pipe to drain after a mispredicted branch. This stall occurs if the corrected target instruction reaches the dispatch stage before the pipe has emptied. See also event D1h.

**Event Select: D3h - Dispatch Stall for Serialization**

The number of processor cycles the decoder is stalled due to a serializing operation, which waits for the execution pipeline to drain. Relatively rare; mainly associated with system instructions. See also event D1h.

**Event Select: D4h - Dispatch Stall for Segment Load**

The number of processor cycles the decoder is stalled due to a segment load instruction being encountered while execution of a previous segment load operation is still pending. Relatively rare except in 16-bit code. See also event D1h.

**Event Select: D5h - Dispatch Stall for Reorder Buffer Full**

The number of processor cycles the decoder is stalled because the reorder buffer is full. May occur simultaneously with certain other stall conditions. See event D1h.

**Event Select: D6h - Dispatch Stall for Reservation Station Full**

The number of processor cycles the decoder is stalled because a required integer unit reservation stations is full. May occur simultaneously with certain other stall conditions. See event D1h.

**Event Select: D7h - Dispatch Stall for FPU Full**

The number of processor cycles the decoder is stalled because the scheduler for the Floating Point Unit is full. This condition can be caused by a lack of parallelism in FP-intensive code, or by cache misses on FP operand loads (which could also show up as event D8h instead, depending on the nature of the instruction sequences). May occur simultaneously with certain other stall conditions. See event D1h.

**Event Select: D8h - Dispatch Stall for LS Full**

The number of processor cycles the decoder is stalled because the Load/Store Unit is full. This generally occurs due to heavy cache miss activity. May occur simultaneously with certain other stall conditions. See event D1h.

**Event Select: D9h - Dispatch Stall Waiting for All Quiet**

The number of processor cycles the decoder is stalled waiting for all outstanding requests to the system to be resolved. Relatively rare; associated with certain system instructions and types of interrupts. May partially overlap certain other stall conditions; see event D1h.

**Event Select: DAh - Dispatch Stall for Far Transfer or Resync to Retire**

The number of processor cycles the decoder is stalled waiting for the execution pipeline to drain before dispatching the target instructions of a far control transfer or a Resync (an instruction stream restart associated with certain microcode assists). Relatively rare; does not overlap with other stall conditions. See also event D1h.

**Event Select: DBh - FPU Exceptions**

The number of floating point unit exceptions for microcode assists. The UNIT\_MASK may be used to isolate specific types of exceptions.

UNIT_MASK	
01h	x87 reclass microfaults
02h	SSE retype microfaults
04h	SSE reclass microfaults
08h	SSE and x87 microtraps

**Event Select: DCh - DR0 Breakpoint Matches**

The number of matches on the address in breakpoint register DR0, per the breakpoint type specified in DR7. The breakpoint does not have to be enabled. Each instruction breakpoint match incurs an overhead of about 120 cycles. Load/store breakpoint matches do not incur any overhead.

**Event Select: DDh - DR1 Breakpoint Matches**

The number of matches on the address in breakpoint register DR1. See notes for event DCh.

**Event Select: DEh - DR2 Breakpoint Matches**

The number of matches on the address in breakpoint register DR2. See notes for event DCh.

**Event Select: DFh - DR3 Breakpoint Matches**

The number of matches on the address in breakpoint register DR3. See notes for event DCh.

**10.2.1.7 Memory Controller Events****Event Select: E0h - DRAM Accesses**

The number of memory accesses performed by the local DRAM controller. The UNIT\_MASK may be used to isolate the different DRAM page access cases. Page miss cases incur an extra latency to open a page; page conflict cases incur both a page-close as well as page-open penalties. These penalties may be overlapped by DRAM accesses for other requests and don't necessarily represent lost DRAM bandwidth. The associated penalties are as follows:

Page miss = Trcd (DRAM RAS-to-CAS delay)

Page conflict =  $T_{rp} + T_{rcd}$  (DRAM row-precharge time plus RAS-to-CAS delay)

Each DRAM access represents one 64-byte block of data transferred if the DRAM is configured for 64-byte granularity, or one 32-byte block if the DRAM is configured for 32-byte granularity. (The latter is only applicable to single-channel DRAM systems, which may be configured either way.)

UNIT_MASK	
01h	Page hit
02h	Page Miss
04h	Page Conflict

### Event Select: E1 - Memory Controller Page Table Overflows

The number of page table overflows in the local DRAM controller. This table maintains information about which DRAM pages are open. An overflow occurs when a request for a new page arrives when the maximum number of pages are already open. Each occurrence reflects an access latency penalty equivalent to a page conflict.

### Event Select: E3 - Memory Controller Turnarounds

The number of turnarounds on the local DRAM data bus. The UNIT\_MASK may be used to isolate the different cases. These represent lost DRAM bandwidth, which may be calculated as follows (in bytes per occurrence):

$\text{DIMM turnaround} = \text{DRAM\_width\_in\_bytes} * 2 \text{ edges\_per\_memclk} * 2$

$\text{R/W turnaround} = \text{DRAM\_width\_in\_bytes} * 2 \text{ edges\_per\_memclk} * 1$

$\text{R/W turnaround} = \text{DRAM\_width\_in\_bytes} * 2 \text{ edges\_per\_memclk} * (T_{cl}-1)$

where DRAM\_width\_in\_bytes is 8 or 16 (for single- or dual-channel systems), and  $T_{cl}$  is the CAS latency of the DRAM in memory system clock cycles (where the memory clock for DDR-400, or PC3200 DIMMS, for example, would be 200 MHz).

UNIT_MASK	
01h	DIMM (chip select) turnaround
02h	Read to write turnaround
04h	Write to read turnaround



**Event Select: E4h - Memory Controller Bypass Counter Saturation**

UNIT_MASK	
01h	Memory controller high priority bypass
02h	Memory controller low priority bypass
04h	DRAM controller interface bypass
08h	DRAM controller queue bypass

**Event Select: E5 - Sized Blocks**

**Sized Read/Write activity:** The Sized Read/Write events reflect 32- or 64-byte transfers (as opposed to other sizes which could be anywhere between 1 and 64 bytes), from either the processor or the Hostbridge (on any node in an MP system). Such accesses from the processor would be due only to write combining buffer flushes, where 32-byte accesses would reflect flushes of partially-filled buffers. Event 65h provides a count of sized write requests associated with WC buffer flushes; comparing that with counts for these events (providing there is very little Hostbridge activity at the same time) will give an indication of how efficiently the write combining buffers are being used. Event 65h may also be useful in factoring out WC flushes when comparing these events with the Upstream Requests component of event ECh.

UNIT_MASK	
04h	32-byte Sized Writes (Revision D and later revisions)
08h	64-byte Sized Writes (Revision D and later revisions)
10h	32-byte Sized Reads (Revision D and later revisions)
20h	64-byte Sized Reads (Revision D and later revisions)

**Event Select: E8h - ECC Errors**

UNIT_MASK	
80h	Number of correctable and Uncorrectable DRAM ECC errors (Revision E)

**Event Select: E9h - CPU/IO Requests to Memory/IO****Revision E**

These events reflect request flow between units and nodes, as selected by the UNIT\_MASK. The UNIT\_MASK is divided into two fields: request type (CPU or I/O access to I/O or Memory) and source/target location (local vs. remote). One or more requests types must be enabled with UNIT\_MASK[3:0], and at least one source and one target location must be selected with UNIT\_MASK[7:4]. Each event reflects a request of the selected type(s) going from the selected source(s) to the selected target(s).

Not all possible paths are supported. The following table shows the UNIT\_MASK values that are valid for each request type.

Source/Target	CPU to Mem	CPU to I/O	I/O to Mem	I/O to I/O
Local -> Local	A8h	A4h	A2h	A1h
Local -> Remote	98h	94h	92h	91h
Remote -> Local	-	64h	-	61h
Remote -> Remote	-	-	-	-

Any of the mask values shown may be logically ORed to combine the events. For instance, local CPU requests to both local and remote nodes would be  $A8h \mid 98h = B8h$ . Any CPU to any I/O would be  $A4h \mid 94h \mid 64h = F4h$  (but remote CPU to remote I/O requests would not be included).

**Note:** It is not possible to tell from these events how much data is going in which direction, as there is no distinction between reads and writes. Also, particularly for I/O, the requests may be for varying amounts of data, anywhere from one to sixty-four bytes. Event E5h provides an indication of 32- and 64-byte read and write transfers for such requests (although from the target point of view). For a direct measure of the amount and direction of data flowing between nodes, use events F6h, F7h and F8h.

UNIT_MASK	
01h	I/O to I/O
02h	I/O to Mem
04h	CPU to I/O
08h	CPU to Mem
10h	To remote node
20h	To local node
40h	From remote node
80h	From local node

## Event Select: EAh - Cache Block Commands

## Revision E

The number of requests made to the system for cache line transfers or coherency state changes, by request type. Each increment represents one cache line transfer, except for Change-to-Dirty. If a Change-to-Dirty request hits on a line in another processor's cache that is in the Owned state, the

request causes a cache line transfer, otherwise no data transfer is associated with Change-to-Dirty requests.

UNIT_MASK	
01h	Victim Block (Writeback)
04h	Read Block (Dcache load miss refill)
08h	Read Block Shared (Icache refill)
10h	Read Block Modified (Dcache store miss refill)
20h	Change to Dirty (first store to clean block already in cache)

### Event Select: EBh - Sized Commands

The number of Sized Read/Write commands handled by the System Request Interface (local processor and hostbridge interface to the system). These commands may originate from the processor or hostbridge. Typical uses of the various Sized Read/Write commands are given in the UNIT\_MASK table. See also event E5h, which covers commonly-used block sizes for these requests, and event ECh, which provides a separate measure of Hostbridge accesses.

UNIT_MASK		Typical Usage
01h	NonPosted SzWr Byte (1-32 bytes)	Legacy or mapped I/O, typically 1–4 bytes
02h	NonPosted SzWr Dword (1-16 dwords)	Legacy or mapped I/O, typically 1 Dword
04h	Posted SzWr Byte (1-32 bytes)	Sub-cache-line DMA writes, size varies; also flushes of partially-filled Write Combining buffer
08h	Posted SzWr Dword (1-16 dwords)	Block-oriented DMA writes, often cache-line sized; also processor Write Combining buffer flushes
10h	SzRd Byte (4 bytes)	Legacy or mapped I/O
20h	SzRd Dword (1-16 dwords)	Block-oriented DMA reads, typically cache-line size
40h	RdModWr	

### Event Select: ECh - Probe Responses and Upstream Requests

This covers two unrelated sets of events — cache probe results and requests received by the Hostbridge from devices on non-coherent links.

**Probe results:** These events reflect the results of probes sent from a memory controller to local caches. They provide an indication of the degree data and code is shared between processors (or moved between processors due to process migration). The dirty-hit events indicate the transfer of a 64-byte cache line to the requestor (for a read or cache refill) or the target memory (for a write). The system bandwidth used by these, in terms of bytes per unit of time, may be calculated as 64 times the event count, divided by the elapsed time. Sized writes to memory that cover a full cache line do not

incur this cache line transfer — they simply invalidate the line and are reported as clean hits. Cache line transfers occur for Change2Dirty requests that hit cache lines in the Owned state. (Such cache lines are counted as Modified-state refills for event 6Ch, System Read Responses.)

**Upstream requests:** The upstream read and write events reflect requests originating from a device on a local non-coherent HyperTransport link. The two read events allow display refresh traffic in a UMA system to be measured separately from other DMA activity. Display refresh traffic will typically be dominated by 64-byte transfers. Non-display-related DMA accesses may be anywhere from 1 to 64 bytes in size, but may be dominated by a particular size such as 32 or 64 bytes, depending on the nature of the devices. Event E5h can provide a measure of 32- and 64-byte accesses by the hostbridge (possibly combined with write combining buffer flush activity from the processor, although that can be factored out via event 65h).

UNIT_MASK	
01h	Probe miss
02h	Probe hit clean
04h	Probe hit dirty without memory cancel (probed by Sized Write or Change2Dirty)
08h	Probe hit dirty with memory cancel (probed by DMA read or cache refill request)
10h	Upstream display refresh reads
20h	Upstream non-display refresh reads
40h	Upstream writes (Revision D and later revisions)

### Event Select: EEh - GART Events

These events reflect GART activity, and in particular allow one to calculate the GART TLB miss ratio as  $\text{GART\_miss\_count} / \text{GART\_aperture\_hit\_count}$ . GART aperture accesses are typically from I/O devices as opposed to the processor and are generally from a 3D graphics accelerator, but can be from other devices when the GART is used as an IO MMU).

UNIT_MASK	
01h	GART aperture hit on access from CPU
02h	GART aperture hit on access from I/O
04h	GART miss

**10.2.1.8 Crossbar Events****10.2.1.9 HyperTransport™ Interface Events****Event Select: F6h - HyperTransport Link 0 Transmit Bandwidth****F7h - HyperTransport Link 1 Transmit Bandwidth****F8h - HyperTransport Link 2 Transmit Bandwidth**

The number of Dwords transmitted (or unused, in the case of Nops) on the outgoing side of the HyperTransport links. The sum of all four subevents (all four UNIT\_MASK bits set) directly reflects the maximum transmission rate of the link. Link utilization may be calculated by dividing the combined Command, Data and Buffer Release count (UNIT\_MASK 07h) by that value plus the Nop count (UNIT\_MASK 08h). Bandwidth in terms of bytes per unit time for any one component or combination of components is calculated by multiplying the count by four and dividing by elapsed time.

The Data event provides a direct indication of the flow of data around the system. Translating this link-based view into a source/target node based view requires knowledge of the system layout (i.e. which links connect to which nodes).

UNIT_MASK	
01h	Command Dword sent
02h	Data Dword sent
04h	Buffer release Dword sent
08h	Nop Dword sent (idle)



# 11 CPUID

Processor feature capabilities and configuration information are provided through the CPUID instruction. Different information is accessed by (1) setting EAX as an index to the registers to be read, (2) executing the CPUID instruction, and (3) reading the results in EAX, EBX, ECX, and EDX. The phrase *CPUID function X* or *CPUID FnX* refers to the CPUID instruction when EAX is preloaded with X.

## **CPUID Fn[8000\_0000.0000\_0000] AMD Authentic Identifier**

Register	Bits	Description
EAX	31:0	Function 0000_0000h returns the largest CPUID standard-function input value supported by the processor implementation: 0000_0005h. Function 8000_0000h returns the largest CPUID extended-function input value supported by the processor implementation: 8000_0019h.
EBX, ECX, EDX	31:0	The 12 8-bit ASCII character codes to create the string "AuthenticAMD". EBX=6874_7541h "h t u A", ECX=444D_4163h "D M A c", EDX=6974_6E65h "i t n e"

## **CPUID Fn[8000\_0001.0000\_0001] EAX Family, Model, Feature Identifiers**

This register provides identical information to Function 3, Offset FCh.

**Family** is an 8-bit value and is defined as: Family[7:0] = ({0000b, BaseFamily[3:0]} + ExtendedFamily[7:0]). E.g. If BaseFamily[3:0]=Fh and ExtendedFamily[7:0]=01h, then Family[7:0]=10h. This document applies only to family 0Fh processors.

**Model** is an 8-bit value and is defined as: Model[7:0] = {ExtendedModel[3:0], BaseModel[3:0]}. E.g. If ExtendedModel[3:0]=Eh and BaseModel[3:0]=8h, then Model[7:0] = E8h. Model numbers vary with product.

Bits	Description
31:28	Reserved.
27:20	<b>ExtendedFamily</b> : 00h.
19:16	<b>ExtendedModel</b> .
15:12	Reserved.
11:8	<b>BaseFamily</b> : Fh
7:4	<b>BaseModel</b> .
3:0	<b>Stepping</b> : processor stepping (revision) for a specific model.

**CPUID Fn[0000\_0001] EBX LocalApicId, LogicalProcessorCount, CLFlush, 8BitBrandId**

Bits	Description
31:24	<b>LocalApicId:</b> Provides the initial APICID (APIC Offset 20h) value in the three LSBs. After Nodelid (Function 0: Offset 60h) has been initialized, changes to APICID do not effect the value of this register.
23:16	<b>LogicalProcessorCount:</b> If CPUID Fn[8000_0001, 0000_0001]_EDX[HTT] = 1, then this field indicates the number of CPU cores in the processor as CPUID Fn8000_0008[NC] + 1. Otherwise, this field is reserved.
15:8	<b>CLFlush:</b> CLFLUSH size in quadwords = 08h.
7:0	<b>8BitBrandId:</b> 8 bit brand ID = 00h. Indicates that the brand ID is in CPUID Fn8000_0001_EBX.

**CPUID Fn[8000\_0001] EBX BrandId Identifier**

Bits	Description
31:16	Reserved.
11:0	<b>BrandId:</b> brand ID.

**CPUID Fn0000\_0001 ECX Feature Identifiers**

These fields are encoded as 1=Feature is supported by the processor; 0=Feature is not supported by the processor.

Bits	Description
31:1	Reserved.
0	<b>SSE3:</b> SSE3 extensions = 1. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 27 for revision information about this feature flag.

**CPUID Fn8000\_0001 ECX Feature Identifiers**

These fields are encoded as 1=Feature is supported by the processor; 0=Feature is not supported by the processor.

Bits	Description
31:2	Reserved.
1	<b>CmpLegacy:</b> core multi-processing legacy mode (setting varies by product). See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 27 for revision information about this feature flag.
0	<b>LahfSahf:</b> LAHF/SAHF instructions =1.



**CPUID Fn[8000\_0001, 0000\_0001] EDX Feature Identifiers**

These fields are encoded as 1=Feature is supported by the processor; 0=Feature is not supported by the processor. The value returned in EDX may be identical or different for Fn0000\_0001 and Fn8000\_0001, as indicated.

Bits	Function	Description
31	0000_0001	Reserved.
	8000_0001	<b>3DNow</b> : 3DNow!™ instructions = 1.
30	0000_0001	Reserved.
	8000_0001	<b>3DNowExt</b> : AMD extensions to 3DNow!™ instructions = 1.
29	0000_0001	Reserved.
	8000_0001	<b>LM</b> : long mode (may vary by product).
28	0000_0001	<b>HTT</b> : hyper-threading technology. This bit qualifies the meaning of CPUID Fn0000_0001_EBX[LogicalProcessorCount]. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors" on page 27 for revision information about this feature flag.
	8000_0001	Reserved.
27	0000_0001	Reserved.
	8000_0001	<b>RDTS</b> : RDTS instruction = 1.
26	0000_0001	<b>SSE2</b> : SSE2 extensions = 1.
	8000_0001	Reserved.
25	0000_0001	<b>SSE</b> : SSE extensions = 1.
	8000_0001	<b>FXSR</b> : FXSAVE and FXRSTOR instruction optimizations = 1. See "Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors" on page 27 for revision information about this feature flag.
24	both	<b>FXSR</b> : FXSAVE and FXRSTOR instructions = 1.
23	both	<b>MMX</b> : MMX™ instructions = 1.
22	0000_0001	Reserved.
	8000_0001	<b>MmxExt</b> : AMD extensions to MMX™ instructions = 1.
21	both	Reserved.
20	0000_0001	Reserved.
	8000_0001	<b>NX</b> : no-execute page protection = 1.
19	0000_0001	<b>CLFSH</b> : CLFLUSH instruction = 1.
	8000_0001	Reserved.
18	both	Reserved.
17	both	<b>PSE36</b> : page-size extensions = 1.
16	both	<b>PAT</b> : page attribute table = 1.
15	both	<b>CMOV</b> : conditional move instructions, CMOV, FCOMI, FCMOV = 1.
14	both	<b>MCA</b> : machine check architecture, MCG_CAP = 1.

Bits	Function	Description
13	both	<b>PGE</b> : page global extension, CR4.PGE = 1.
12	both	<b>MTRR</b> : memory-type range registers = 1.
11	0000_0001	<b>SysEnterSysExit</b> : SYSENTER and SYSEXIT instructions = 1.
	8000_0001	<b>SysCallSysRet</b> : SYSCALL and SYSRET instructions = 1.
10	both	Reserved.
9	both	<b>APIC</b> : advanced programmable interrupt controller (APIC) exists and is enabled. This bit reflects the state of the APIC Base Address register (APIC_BAR) MSR 0000_001B[ApicEn].
8	both	<b>CMPXCHG8B</b> : CMPXCHG8B instruction = 1.
7	both	<b>MCE</b> : machine check exception, CR4.MCE = 1.
6	both	<b>PAE</b> : physical-address extensions (PAE) = 1.
5	both	<b>MSR</b> : AMD model-specific registers (MSRs), with RDMSR and WRMSR instructions = 1.
4	both	<b>TSE</b> : time stamp counter, RDTSC/RDTSCP instructions, CR4.TSD = 1.
3	both	<b>PSE</b> : page-size extensions (4 MB pages) = 1.
2	both	<b>DE</b> : debugging extensions, IO breakpoints, CR4.DE = 1.
1	both	<b>VME</b> : virtual-mode enhancements = 1.
0	both	<b>FPU</b> : x87 floating point unit on-chip = 1.

### **CPUID Fn8000\_000[4, 3, 2] Processor Name String Identifier**

These return the ASCII string corresponding to the processor name, stored in MSR C001\_00[35:30]. The MSRs are mapped to these registers as follows:

Function 8000\_0002: {EDX, ECX, EBX, EAX} == {MSR C001\_0031, MSR C001\_0030};

Function 8000\_0003: {EDX, ECX, EBX, EAX} == {MSR C001\_0033, MSR C001\_0032};

Function 8000\_0004: {EDX, ECX, EBX, EAX} == {MSR C001\_0035, MSR C001\_0034};

### **CPUID Fn8000\_0005 TLB and L1 Cache Identifiers**

This provides the processor's first level cache and TLB characteristics for each CPU core. The associativity fields returned are encoded as follows:

00h Reserved.

01h Direct mapped.

02h - FEh Specifies the associativity; e.g., 04h would indicate a 4-way associativity.

FFh Fully associative.

Register	Bits	Description
EAX	31:24	Data TLB associativity for 2 MB and 4 MB pages = FFh.

Register	Bits	Description
EAX	23:16	Data TLB number of entries for 2 MB and 4 MB pages = 8. The value returned is for the number of entries available for the 2 MB page size; 4 MB pages require two 2 MB entries, so the number of entries available for the 4 MB page size is one-half the returned value.
EAX	15:8	Instruction TLB associativity for 2 MB and 4 MB pages = FFh.
EAX	7:0	Instruction TLB number of entries for 2 MB and 4 MB pages = 8. The value returned is for the number of entries available for the 2 MB page size; 4 MB pages require two 2 MB entries, so the number of entries available for the 4 MB page size is one-half the returned value.
EBX	31:24	Data TLB associativity for 4 KB pages = FFh.
EBX	23:16	Data TLB number of entries for 4 KB pages = 32.
EBX	15:8	Instruction TLB associativity for 4 KB pages = FFh.
EBX	7:0	Instruction TLB number of entries for 4 KB pages = 32.
ECX	31:24	L1 data cache size in KB = 64.
ECX	23:16	L1 data cache associativity = 2.
ECX	15:8	L1 data cache lines per tag = 1.
ECX	7:0	L1 data cache line size in bytes = 64.
EDX	31:24	L1 instruction cache size in KB = 64.
EDX	23:16	L1 instruction cache associativity = 2.
EDX	15:8	L1 instruction cache lines per tag = 1.
EDX	7:0	L1 instruction cache line size in bytes = 64.

### **CPUID Fn8000\_0006 L2 Cache Identifiers**

This provides the processor's second level cache and TLB characteristics for each CPU core.

The presence of a unified L2 TLB is indicated by a value of 0000h in the upper 16 bits of the EAX and EBX registers. The unified L2 TLB information is contained in the lower 16 bits of these registers.

The associativity fields are encoded as follows:

- |                                      |                         |
|--------------------------------------|-------------------------|
| 0h: The L2 cache or TLB is disabled. | 6h: 8-way associative.  |
| 1h: Direct mapped.                   | 8h: 16-way associative. |
| 2h: 2-way associative.               | Fh: Fully associative.  |
| 4h: 4-way associative.               |                         |

All other encodings are reserved.

Register	Bits	Description
EAX	31:0	The L2 TLB has no entries for 2 MByte and 4 MByte pages. 0000_000h
EBX	31:24	L2 data TLB associativity for 4 KB pages = 4.
EBX	23:16	L2 data TLB number of entries for 4 KB pages = 512.

Register	Bits	Description
EBX	15:8	L2 instruction TLB associativity for 4 KB pages = 4.
EBX	7:0	L2 instruction TLB number of entries for 4 KB pages = 512.
ECX	31:24	L2 cache size in KB (varies with product). May be one of 128, 256, 512, or 1024.
ECX	23:16	L2 cache associativity = 8.
ECX	15:8	L2 cache lines per tag = 1.
ECX	7:0	L2 cache line size in bytes = 64.
EDX	31:0	Reserved.

### **CPUID Fn8000 0007 Advanced Power Management Information**

This provides advanced power management feature identifiers. Unless otherwise specified, these bits are encoded as 1=Feature is supported by the processor; 0=Feature is not supported by the processor.

Register	Bits	Description
EAX, EBX, ECX	31:0	Reserved.
EDX	31:4	Reserved.
EDX	3	<b>TTP</b> : THERMTRIP is supported = 1.
EDX	2	<b>VID</b> : Voltage ID control is supported.
EDX	1	<b>FID</b> : Frequency ID control is supported.
EDX	0	<b>TS</b> : Temperature sensor = 1.

### **CPUID Fn8000 0008 Address Size And Physical Core Count Information**

This provides information about the number of physical CPU cores and the maximum physical and linear address width supported by the processor.

Register	Bits	Description
EAX	31:16	Reserved.
EAX	15:8	<b>LinAddrSize</b> : Maximum linear byte address size in bits. If the processor supports long mode (see CPUID Fn[8000_0001, 0000_0001]_EDX[LM]) then this is 30h; else this is 20h.
EAX	7:0	<b>PhysAddrSize</b> : Maximum physical byte address size in bits = 28h.
EBX	31:0	Reserved.
ECX	31:8	Reserved.
ECX	7:0	<b>NC</b> : number of physical CPU cores - 1. The number of CPU cores in the processor is NC+1 (e.g., if NC=0, then there is one CPU core).
EDX	3:0	Reserved.

**CPUID Fn8000\_00[18:09] Reserved**

These return all zeros in EAX, EBX, ECX, and EDX.



## 12 BIOS Checklist

---

The checklist in this chapter reviews information that is described elsewhere and is required by BIOS developers to properly incorporate AMD Athlon™ 64 and AMD Opteron™ Processors into systems.

### 12.1 CPUID

Use the CPUID instruction to properly identify the processor.

- Determine the processor type, stepping, and available features by using functions 0000\_0001h and 8000\_0001h of the CPUID instruction.
- Boot-up display - The processor name should be displayed according to the processor name string defined in *CPUID Guide for the AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25481.

For more information on the CPUID instruction, see *Chapter 11, "CPUID", AMD Processor Recognition Application Note*, order# 20734 and *CPUID Guide for the AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25481.

### 12.2 CPU Speed Detection

The BIOS can use the time-stamp counter (TSC) to clock a timed operation and compare the result to the Real-Time Clock (RTC) to determine the operating frequency. See the example of frequency-determination assembler code available on the AMD web site.

### 12.3 HyperTransport™ Link Detection

After any reset, the BIOS should check every HyperTransport™ link on the processor to detect whether the link is connected by executing the following steps:

1. Wait until the LinkConPend bit in Function 0, Offset 98h, B8h, or D8h is cleared.
2. Check LinkCon bit in Function 0, Offset 98h, B8h, or D8h. If LinkCon bit has a value of 0, the link is not connected. If LinkCon bit has a value of 1, the link is connected.

After a warm reset, if the link is connected, the BIOS should additionally check LinkFail bit in Function 0, Offset 84h, A4h, or C4h to determine if sync flood packets were detected by the link receivers before the warm reset assertion. The LinkFail bit should then be cleared.

## 12.4 HyperTransport™ Link Frequency Selection

The BIOS should determine the set of frequencies that a processor is capable of for every HyperTransport™ link connected to the processor. Each frequency in the set must be defined by the corresponding link frequency capability bit in the LnkFreqCap field, Function 0, Offsets 88h, A8h, C8h, and it must be equal to or less than the maximum frequency values specified in the processor data sheets.

If a HyperTransport™ link is non-coherent, then the BIOS should initialize the HyperTransport™ link frequency (Link field, Function 0, Offsets 88h, A8h, C8h) with the highest value from the set of frequencies of which the processor is capable and which is less than or equal to the maximum frequency values specified in the chipset data sheets, and the maximum frequency supported by the platform.

If a HyperTransport™ link is coherent, then the BIOS should initialize the HyperTransport™ link frequencies (Link field, Function 0, Offsets 88h, A8h, C8h) of both processors with the highest common frequency value from the sets of frequencies of which each processor is capable that is less than or equal to the maximum frequency supported by the platform.

## 12.5 Multiprocessing Capability Detection

The multiprocessing capability of the AMD Opteron™ processor is determined by the MPCap and BigMPCap bits in the Northbridge Capability Register (Function 3, Offset E8h).

Multiprocessing Capability	MPCap	BigMPCap
UP Capable	0	0
DP Capable	1	0
MP Capable	1	1

During POST, the BIOS checks the multiprocessing capability of AMD Opteron™ processors, and configures the system accordingly.

MP setup by the BIOS is also required in a UP where dual-cores are present.

Multiprocessing capability detection is not required in a UP system.

All processors must be DP capable or MP capable in a DP system. If any processor is only UP capable, the BIOS must configure the BSP as a UP processor, and must not initialize the AP.

All processors must be MP capable in an MP system. If any processor is not MP capable, the BIOS must configure the BSP as a UP processor, and must not initialize APs.

If all processors do not have adequate multiprocessing capability for a DP or an MP system, the BIOS must display the following message:

```
***** Warning: non-MP Processor *****
```



The processor(s) installed in your system are not multiprocessing capable. Now your system will halt.

If all processors have adequate multiprocessing capability for a DP or an MP system, but have different model numbers or operate at different frequencies, the BIOS can do one of the following:

- configure the BSP as a UP processor, not initialize APs, and display a message indicating that processors with different model numbers or maximum frequencies have been installed, or
- implement the algorithm defined in “Processors With Different Revisions or Maximum Frequencies in Multiprocessor Systems”.

## 12.6 Setup for Dual Core Processors

To ensure that dual core processors work correctly in existing operating systems there are several steps that the BIOS must follow. These steps must only be performed on a dual core processor when both cores are enabled.

1. Re-direct all MC4 accesses and error logging to core 0. This is done by setting the NbMcaToMstCpuEn bit (Function 3, Offset 44h, bit 27).

Setting this bit is required to ensure that current operating systems do not become confused with conflicting programming of this shared resource between two cores.

2. Ensure that the core number is the least significant bit of the initial APIC ID. This is done by setting the InitApicIdCpuIdLo bit (MSR C001\_001Fh, bit 54) in each core.

Setting this bit is required because current operating systems expect the core ID bits to be in the least significant bit positions of the APIC ID. The operating system makes scheduling decisions based on this information.

3. The firmware must properly report dual-core processors in a number of tables required by the Advanced Configuration and Power Interface (ACPI) specification:
  - The Static Resource Affinity Table (SRAT) must contain a "Processor Local APIC/SAPIC Affinity Structure" for each core rather than for each processor. Cores within the same processor must have the same proximity domain.
  - The CPU objects in the Differentiated System Description Table (DSDT) must contain the correct number of cores.
  - The Multiple APIC Description Table (MADT) must contain a "Processor Local APIC" record and an "ACPI Processor object" for each core rather than for each processor.
4. Each core, rather than each processor, should have a processor entry in the MP configuration table as defined in the MultiProcessor Specification ver, 1.4.
5. Software can always calculate the number of physical processors in a system by taking the total number of processor entries/records in the MADT or MP configuration tables and dividing by the number of cores per processor returned by CPUID Fn8000\_0008.

6. The order of the processor entries/records in the MADT and the MP configuration table are not specified by AMD. However, some firmware vendors may choose to work around the licensing restrictions of some legacy operating systems by listing the entries/records of the first core of each processor (core 0) before listing the entries/records of the remaining cores:

```
processor 0, core 0
...
processor N, core 0
processor 0, core 1
...
processor N, core 1
```

## 12.7 APIC ID Assignment Requirements

The arbitration protocol used when sending data over the original serial APIC bus required that IOAPIC IDs be unique and not overlap the processor APIC IDs (i.e. mapped into the same linear ID space such as APIC IDs from 0 to N and IOAPIC IDs from N+1 to N+M). A number of operating systems may still have this requirement even though AMD NPT processors do not require it. Most of these operating systems also support numbering the IOAPIC IDs first before numbering the processor APIC IDs. Firmware should support this model for the broadest operating system support.

If the total number of IOAPICs plus the total number of processor APICs (number of processor \* number of cores per processor) is less than 16:

- $APIC\_ID[node=i] = i$  for a single-core system
- $APIC\_ID[node=i, core=j] = i*2 + j$  for a dual-core system
- APIC\_ID for IOAPIC starts from the next available number after the last APIC\_ID assigned to the processor.

If the total number of processor APICs is equal to or greater than 16:

- APIC\_ID for IOAPIC starts from 0
- $APIC\_ID[node=i] = i + IOAPIC\_OFFSET$  for a single-core system
- $APIC\_ID[node=i, core=j] = (i*2 + j) + IOAPIC\_OFFSET$  for a dual-core system
- $IOAPIC\_OFFSET = \text{total number of IOAPIC, or total number of IOAPIC} + 1$  if it is an odd number

## 12.8 Processors With Different Revisions or Maximum Frequencies in Multiprocessor Systems

AMD Opteron™ processors with different revisions or maximum frequencies can be mixed in a multiprocessor system if each processor is C0 or a later silicon revision. Refer to the *Revision Guide for AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25759 for information about processors that can be mixed in a multiprocessor system.

The following requirements must be met when processors of different revisions or maximum frequencies are mixed:

1. The system must be configured as a DP or an MP system. See section “Multiprocessing Capability Detection” on page 328 for more information.
2. All processors must properly initialize Link field in the corresponding Function 0, Offset 88h, A8h, or C8h register. See “HyperTransport™ Link Frequency Selection” on page 328 for more information.
3. The BIOS must set the FID of all processors in the system to the same value. That value should be the highest common FID for all processors in the system. The set of FID values of which each processor is capable is determined as follows:

```
Execute CPUID instruction with EAX=80000007h;
if(EDX[1] == 1) // Processor supports FID changes.
    Read MSR C001_0042h; // Get FIDVID Status register.
    MaxFID = EAX[21:16];
    // MaxFID is the maximum FID of which the processor is capable.
    // The processor is capable of all FIDs from MaxFID down to the FID that
    // represents 1600 MHz, and if any FID is a portal core frequency, then
    // the corresponding minimum core frequency is also a FID of which the
    // processor is capable. See Table 69.
else // Processor does not supports FID changes.
    Read MSR C001_0015h; // Get HWCR register.
    START_FID = EAX[29:24];
    // This is the only FID of which the processor is capable.
```

The BIOS should determine whether a highest common FID value for all processors in the system exists. If such a value exists, then FID values for all processors must be set to that value. See Chapter 9, “Power and Thermal Management” for more information on FID changes. If there is not a common FID value for all processors in the system, then the BIOS should not change the FID on any processor, and it should display a warning message.

Power management registers defined in Table 78 must be programmed before changing the processor FID value.

4. The BIOS must follow P-state change rules specified in Chapter 9, “Power and Thermal Management”, however, VID changes do not have to be made.

5. All processors must have the same L2 cache size, returned in ECX[31:16] when a CPUID instruction is executed with EAX=80000006h. If all processors in the system do not have the same L2 cache size, then the BIOS should display a warning message. See *CPUID Guide for the AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25481 for more information.
6. The BIOS must ensure that all CPUs present the same feature set to the operating system. For instance, if a CPU in the system does not support SSE3 instructions, then all CPUs in the system must be configured to not report the SSE3 capability to OS or application software.

To accomplish this, BIOS must first determine the feature set that is common to all CPUs in the system. Then, for all CPUs, BIOS must disable any features that are not supported by all CPUs. The following registers contain the feature set bits:

- CPUIDFeatures (MSR C001\_1004h)
- CPUIDExtFeatures (MSR C001\_1005h)

The bits in these registers identify individual features. A value of 1 indicates that the feature is supported and a value of 0 indicates that the feature is not supported. All of these bits are read/write, thus, if a bit has a value of 1, it can be written to a 0 to make it appear to the OS that the feature is not supported.

7. The errata, workarounds, and microcode updates must be properly applied to each processor in the system. See *Revision Guide for AMD Athlon™ 64 and AMD Opteron™ Processors*, order# 25759 for more information.
8. If S3 power state is implemented in the system, all steps listed above must be executed on a resume from S3.

## 12.9 Model-Specific Registers (MSRs)

Access only the MSRs that are implemented in the processor.

- Follow the processor MSR programming sequence described in this document. This includes setting the HWCR, SYSCFG, MTRRs, IORRs, clock control MSR, Top of Memory, and other registers.

### 12.9.1 Software Considerations for Accessing Northbridge MSRs in Dual Core Processors

MSRs that control Northbridge functions are shared between both cores in a dual core processor. If control of Northbridge functions is shared between software on both cores, software must ensure that only one core at a time is allowed to access the shared MSR.

**Table 80. Northbridge MSRs**

Address	Register Name	Description
0410h	MC4_CTL	page 213
0411h	MC4_STATUS	page 213
0412h	MC4_ADDR	page 213
C001_001Fh	NB_CFG	page 373
C001_0041h	FIDVID_CTL	page 382
C001_0042h	FIDVID_STATUS	page 384
C001_0048h	MCi_CTL_MASK	page 213

## 12.10 Machine Check Architecture (MCA)

The processor supports a machine check architecture that allows reporting and handling of system errors through a consistent interface.

- The global MCA Status register must be cleared (all machine check features disabled).
- MC0\_CTL must be set to all 1s because some do not set these bits.
- In addition, the local MCA status and address registers for the following five reporting blocks must also be cleared (all error and parity bits disabled):
  - BU—Bus Unit
  - IC—Instruction Cache Unit
  - DC—Data Cache Unit
  - LS—Load Store Unit
  - NB—Northbridge Unit

After reset, the BIOS needs to determine if the cause of the reset was a power-on or soft reset.

- When power-on reset occurs, the BIOS must clear the MCA registers.
- Otherwise, the BIOS should check for any MCA information that may be related to the soft reset.

For more information on MCA, see Chapter 5, “Machine Check Architecture.”

### 12.10.1 GART Table Walk Error Reporting

The GART table walk error reporting function is enabled by setting bit 10 of the MC4\_CTL register (MSR 410h). This results in a machine check exception when an AGP aperture access has no matching translation in the GART. This error is typically caused by a software graphics driver that

improperly reserves or allocates aperture pages in the GART, resulting in benign visual artifacts which are often undetected on other platforms. Setting MC4\_CTL[10] allows software developers to debug this error; the resulting benign machine check errors can, however, confuse an end user. For this reason, AMD recommends that the BIOS developers disable this function by setting bit 10 of MC4\_CTL\_MASK register (MSR C001\_0048h) to a value of 1. This bit must be set before MC4\_CTL[10] bit is set. AMD also recommends adding a setup option to the BIOS setup menu. The following should be displayed in the setup option:

Gart Table Walk Error MC reporting:      Disabled/Enabled.

The default setting is disabled. The device driver developer may enable this function for implementation and testing purposes. Also, a help message should be added with this setup option. An example of the help message is:

This option should remain disabled for normal operation.

## 12.11 GART Register Restoration After S3 Resume

After a resume from S3, GartEn (Function 3, Offset 90h) must be restored after all GART related and AGP device registers are restored.

## 12.12 Register Settings in UMA Systems

AMD recommends that UMA graphics systems use the following register field values:

Register and Field Position	Field Name	Value
Function 0, Offset 68h, [28]	EnCpuRdHiPri, enable high priority CPU reads	0b
Function 0, Offset 68h, [27:26]	HiPriBypCnt, high-priority bypass count	11b
Function 0, Offset 68h, [11]	RspPassPW, response pass posted writes	1b
Function 2, Offset 94h, [19]	DCC_EN, dynamic idle cycle counter enable	1b
Function 2, Offset 90h, [27:25]	BypMax, bypass maximum	111b
Function 2, Offset 90h, [19]	32ByteEn, enable 32-byte granularity	1b
Function 2, Offset 90h, [15:14]	RdWrQByp, read/write queue bypass count	11b
MSR C001_001Fh, [36]	DisDatMsk, disable data mask	0b
MSR C001_001Fh, [9]	EnRefUseFreeBuf, enable display refresh to use free list buffers	Revision CG and earlier 1b Revision D and later 0b

Refer to Table 37 for information on recommended flow control buffer allocation in UMA graphics systems.

## 12.13 Memory Map

### 12.13.1 I/O and Memory Type and Range Registers (IORRs, MTRRs)

The memory-type and range registers control the access and cacheability of memory regions in the processor.

- Follow the processor MTRR and IORR programming sequence that is described in this document.
- To initialize MTRRs in BIOS, set the default memory cache type for all addresses to uncacheable, then use a variable range MTRR to set all physical memory as cacheable (writeback), and finally use the fixed-range MTRRs to handle the special cases in the 640-Kbyte–1-Mbyte region. MTRRs should be modified while cache is disabled. Modifying MTRRs while cache is enabled will result in undefined behavior.
- The GART driver (miniport) should program the AGP aperture.
- The BIOS does not map the cacheability of the video frame buffer and AGP aperture space; it is done by the operating system, video, and AGP drivers.

### 12.13.2 Memory Map Registers (MMRs)

The processor contains several memory map registers to support the appropriate software view of memory.

- Program the correct values in the MSRs, such as Top of Memory (TOP\_MEM).

## 12.14 Access Routing And Type Determination

### 12.14.1 Memory Access to the Physical Address Space

All memory accesses to the physical address space from a CPU core are sent to its associated northbridge (NB). All memory accesses from an IO link are routed through the NB. An IO link access to physical address space indicates to the NB the cache attribute (Coherent or NonCoherent).

A CPU core access to physical address space has two attributes that the CPU must determine before issuing the access to the NB: the cache attribute (e.g., WB, WC, UC) and the access destination (DRAM or MMIO).

#### 12.14.1.1 Determining The Cache Attribute

1. The CPU translates the logical address to a physical address. In that process it determines the initial cache attribute based on the settings of the Page Table Entry PAT bits, the MTRR Default Memory Type Register (MSR 02FF), the Variable-Size MTRRs (MSRs 02[0F:00]), and the Fixed-Size MTRRs (MSRs 02[6F:68, 59, 58, 50]).

2. The ASeg and TSeg SMM mechanisms are then checked in parallel to determine if the initial cache attribute should be overridden (see MSR C001\_0112 and MSR C001\_0113). If the address falls within an enabled ASeg/TSeg region, then the final cache attribute is determined as specified in MSR C001\_0113.

This mechanism is managed by the BIOS and does not require any setup or changes by system software.

#### 12.14.1.2 Determining The Access Destination for CPU Accesses

The access destination is based on the highest priority of the following ranges that the access falls in:

1. (Lowest priority) Compare against the top-of memory registers (MSR C001\_001A and MSR C001\_001D).
2. The Fixed-Size MTRRs (MSRs 02[6F:68, 59, 58, 50]).
3. The IORRs (MSRs C001\_00[18, 16] and MSRs C001\_00[19, 17]).
4. TSEG & ASEG (MSR C001\_0112 and MSR C001\_0113).
5. (Highest priority) NB AGP aperture range registers.

To determine the access destination, the following steps are taken:

1. The CPU compares the address against the Top Of Memory Register (MSR C001\_001A), and the Top Of Memory 2 Register (MSR C001\_001D), to determine if the default access destination is DRAM or MMIO space.
2. For addresses below 1M byte, the address is then compared against the appropriate Fixed MTRRs (MSRs 02[6F:68, 59, 58, 50]) to override the default access destination. Each fixed MTRR includes two bits, RdDrAm and WrDrAm, that determine the destination based on the access type.
3. The CPU then compares the address against the IORRs (MSRs C001\_00[18, 16] and MSRs C001\_00[19, 17]); if it matches, the default access destination is overridden as specified by the IORRs. BIOS can use the IORRs to create an IO hole within a range of addresses that would normally be mapped to DRAM. It can also use the IORRs to re-assert a DRAM destination for a range of addresses that fall within a bigger IO hole that overlays DRAM. Some key points to consider:
  - a. Operating system software never needs to program IORRs to re-map addresses that naturally target DRAM; any such programming is done by the BIOS.
  - b. The IORRs should not cover the range used for the AGP aperture if the GART logic in the NB is enabled.
  - c. The IORRs should be programmed to cover the AGP aperture if the aperture/GART translation is handled by an IO device (e.g., the chipset).
4. The ASeg and TSeg SMM mechanisms are then checked in parallel to determine if the destination should be overridden (see MSR C001\_0112 and MSR C001\_0113). If the address falls within an enabled ASeg/TSeg region, then the destination is determined as specified in MSR C001\_0113.



This mechanism is managed by the BIOS and does not require any setup or changes by system software.

## 12.14.2 Northbridge Processing of Accesses To Physical Address Space

The physical address, its cacheability type, its access type and its access destination is presented to the processor NB for further processing. Processing of an access is performed by NB in the following manner:

1. Regardless of the access destination, if supplied, the physical address is checked against the NB's AGP-aperture range-registers, if enabled; if the address matches, the NB translates the physical address through the AGP GART. A match in the AGP aperture overrides any match to the NB's DRAM Base/Limit Registers, Function 1, Offsets [7C:40], and Memory Mapped IO Base/Limit Registers, Function 1, Offsets [BC:80].
2. For accesses from I/O devices, the cacheability attribute from the GART entry (GartPteCoh bit) is applied; for accesses from a CPU, the attribute already applied by the CPU is used and the GartPteCoh bit is ignored. (System software should ensure that the cacheability attribute assigned to an AGP aperture matches the GartPteCoh bit in the matching GART entry.)
3. Accesses that do not match the AGP aperture and post-GART translated addresses are compared against the NB DRAM Base/Limit Registers, Function 1, Offsets [7C:40], and Memory Mapped IO Base/Limit Registers, Function 1, Offsets [BC:80], to determine the destination link of the access.
4. For accesses from a CPU, the access destination determines whether to compare the address against the DRAM Base/Limit Registers, Function 1, Offsets [7C:40] or the Memory Mapped IO Base/Limit Registers, Function 1, Offsets [BC:80]. For accesses from an IO device both sets of registers are checked; if an access matches against both, the MMIO registers take precedence.

## 12.15 Error Handling

Error detection and signaling capabilities of all system components: processors, chipsets, BIOS and operating system must be considered when selecting error handling mechanisms.

### 12.15.1 Error Handling Mechanisms

Sync flooding and machine check exception handling mechanisms are described.

#### 12.15.1.1 Sync Flooding

Sync flooding is a HyperTransport™ method used to stop data propagation in the case of a serious error. The device that detects the error initiates sync flood, and all devices in the system that detect sync flood packets on its HyperTransport™ receivers cease normal operation and start transmitting sync flood packets. Sync flood packets will reach the I/O hub. If the I/O hub is programmed to generate a warm reset in the case of sync flood detection, the BIOS can analyze all error bits in the

system that were not cleared by the warm reset and detect the error source. If the I/O hub is not programmed to generate a warm reset, the system will hang.

### **12.15.1.2 Machine Check Exceptions**

Machine check exceptions are generated when uncorrectable error conditions are detected by AMD Athlon™ 64 and AMD Opteron™ processors. For revision E and later revisions certain correctable error conditions generate machine check exceptions. Detected error conditions are caused by the processor or an external I/O device. Machine check exception architecture can be programmed to generate sync flood instead of a machine check exception for some types of uncorrectable errors. See Chapter 5, “Machine Check Architecture” for more information.

## **12.15.2 Errors and Recommended Error Handling**

### **12.15.2.1 CRC Errors on HyperTransport™ Links**

AMD Athlon™ 64 and AMD Opteron™ Processors can be programmed to generate sync flood when a CRC error is detected on a HyperTransport™ link. Sync flooding is enabled with Function 0, Offset 84h, A4h, or C4h, CRCFloodEn bit.

### **12.15.2.2 I/O Errors**

Target aborts on reads and nonposted writes are indicated by ERROR=1 and NXA=0 in their response packets in the HyperTransport™ protocol. If detected by the processor, the recommended error handling mechanism is machine check exception. The BIOS should set Function 3, Offset 40h, TgtAbtEn bit.

Master aborts on reads and nonposted writes are indicated by ERROR=1 and NXA=1 in their response packets in the HyperTransport™ protocol. If detected by the processor, the recommended error handling mechanism is machine check exception. The BIOS should set Function 3, Offset 40h, MstrAbtEn bit.

It is also recommended to set DisPciCfgCpuErrRsp and CpuErrDis bits in Function 3, Offset 44h.

Error status is not returned to the processor for target and master aborts on posted writes from the processor to I/O devices, and the I/O device should respond. Error handling is chipset dependent in the case of target and master aborts on posted writes, PERR and SERR. Chipsets can implement sync flooding, interrupts, and other methods for error handling.

### **12.15.2.3 Machine Check Architecture Errors**

It is recommended that all uncorrectable errors detected by the MCA banks in AMD Athlon™ 64 and AMD Opteron™ Processors with the exception of uncorrectable ECC errors detected by the NB are handled by the machine check exceptions.

#### 12.15.2.4 Uncorrectable ECC Errors Detected by the NB

In AMD Opteron™ multiprocessor system, sync flooding is the recommended response to uncorrectable ECC errors detected by the NB on reads from the DRAM. BIOS must take the following steps:

1. Initialize DRAM on all processors in the system (see Chapter 4, “DRAM Configuration”).
2. Enable ECC checking in all DRAM controllers (Function 3, Offset 44h, bit EccEn), and initialize every DRAM location. ECC errors generated during DRAM initialization should be ignored.
3. Enable sync flooding on uncorrectable ECC errors (Function 3, Offset 44h, bit SyncOnUcEccEn).
4. Enable uncorrectable ECC reporting (Function 3, Offset 40h, bit UnCorrEccEn).
5. Enable the I/O hub to generate a warm reset when it detects sync flood packets on the HyperTransport™ link.
6. Determine the type of reset shortly after boot (Function 0, Offset 6Ch, ColdRstDet bit). If the reset was warm, check the LinkFail bit (Function 0, Offset 84h, A4h, or C4h) for every connected link (see “HyperTransport™ Link Detection” on page 327). If the LinkFail bit is set, sync flood packets were detected by the link before the warm reset. The LinkFail bit should be cleared by writing a 1, and the BIOS should determine the error source. Uncorrectable ECC errors detected on DRAM reads are reported in the MCA NB reporting bank. MCA NB status and address registers (Function 3, Offsets 48h, 4Ch, 50h, and 54h) should be analyzed to determine the DIMM where the error originated.

#### 12.15.2.5 Watchdog Timeout Errors

In AMD Opteron™ and AMD Athlon™ 64 systems sync flooding is the recommended response to watchdog timeout errors. BIOS must take the following steps:

1. Enable sync flooding on watchdog timeout errors (Function 3, Offset 44h, bit SyncOnWdogEn).
2. Enable watchdog timeout reporting (Function 3, Offset 40h, bit WchDogTmrEn).
3. Enable the I/O hub to generate a warm reset when it detects sync flood packets on the HyperTransport™ link.
4. Determine the type of reset shortly after boot (Function 0, Offset 6Ch, ColdRstDet bit). If the reset was warm, check the LinkFail bit (Function 0, Offset 84h, A4h, or C4h) for every connected link (see “HyperTransport™ Link Detection” on page 327). If the LinkFail bit is set, sync flood packets were detected by the link before the warm reset. The LinkFail bit should be cleared by writing a 1, and the BIOS should determine the error source. Watchdog timeout errors are reported in the MCA NB reporting bank. BIOS should not clear the MCi\_Status registers when a watchdog timeout is detected.

## 12.16 Cache Initialization For Temporary Storage During Boot

Prior to initializing the DRAM controller for system memory, the processor cache subsystem may be initialized for use by BIOS as temporary storage. The following steps should be taken to set up the cache in the boot core of the BSP to supply 64-Kbytes of memory, including a stack:

- Clear all the MTRRs.
- Set MSR C001\_0010 [MtrrFixDramEn, MtrrFixDramModEn, and MtrrVarDramEn].
- Enable cache through CR0.
- Determine the base address (BASE) of the 64K memory range to be used.
- Use MTRRs to specify the write-back attribute (WB) to a 64 kilobyte portion of the memory map, starting at BASE.
- Use MTRRs or IORRs to specify the 64 kilobyte portion of the memory map starting at BASE as IO by clearing the RdDram and WrDram bits.
- Read exactly 64 kilobytes of memory starting at BASE, using the REP MOV instruction.
- Set SS=BASE/16.
- Set SP=BASE+64K-4.

Temporary data stored in the cache during boot cannot be written back to DRAM after enabling the DRAM controller using a CLFLUSH or WBINVB instruction. The cache should be invalidated after the DRAM controller is initialized with an INVD instruction.

## 12.17 Cache Testing and Programming

- BIOS must correctly maintain the tag parity and data ECC enable bits.

## 12.18 Memory System Configuration Registers

Node configuration space contains registers that define RAM, PCI memory, and x86 I/O address paths.

- Address mapping registers are contained in function 1 of each node.
- Cold boot and reset sends code fetch and other addresses to the compatibility link of Node0. At cold boot and reset, these mapping registers are disabled.

- Once enabled, these registers distribute addresses according to register mappings, which include node and link routing. See Chapter 7, “HyperTransport™ Technology Configuration and Enumeration.”
- Thereafter, carefully set registers to reflect current needs for code fetch and memory access, i.e., do not map BIOS E0000h and F0000h space to DRAM until BIOS exits execution from the ROM chip. Otherwise, the system will hang. Likewise, memory-mapped I/O space should not be mapped above 1 Mbyte until BIOS exits execution from the ROM chip, or else the system will hang.
- Memory-mapped I/O (MMIO) and PCI I/O mappings must include address space needed to move option ROM code to RAM, as well as the required run-time resources. This could require setting and resetting the MMIO mappings during enumeration and device initialization. Configuration space access of a device with or without an option ROM is defined by the HyperTransport technology configuration mapping.
- Multihost bus multiprocessor systems require that MMIO mappings direct relevant addresses to the buses of devices through the node/link path to the bus.
- Application processors and the bootstrap processor in multiprocessor systems must be mapped separately with paths to target address space.

## 12.19 Processor ID

ACPI 2.0 specifies that each processor is required to have a unique ProcessorID, that although arbitrary, must match its corresponding ACPI Processor ID field in the Processor Local APIC table.

## 12.20 XSDT Table

The Extended System Description Table (XSDT) table defined in ACPI 2.0 must be implemented in the BIOS to support 64-bit operating systems for AMD Athlon™ 64 and AMD Opteron™ Processor based systems.

## 12.21 Detect Target Operating Mode Callback

The operating system notifies the BIOS what the expected operating mode is with the Detect Target Operating Mode callback (INT 15, function EC00h). Based on the target operating mode, the BIOS can enable or disable mode specific performance and functional optimizations that are not visible to system software.

This callback does not change the operating mode; it only declares the target mode to the BIOS. It should be executed only once by the BSP before the first transition into long mode.

The default operating mode assumed by the BIOS is Legacy Mode Target Only. If this is not the target operating mode, system software must execute this callback to change it before transitioning to long mode for the first time. If the target operating mode is Legacy Mode Target Only, the callback does not need to be executed.

The Detect Target Operating Mode callback inputs are stored in the AX and BL registers. AX has a value of EC00h, selecting the Detect Target Operating Mode function. One of the following values in the BL register selects the operating mode:

- 01h — Legacy Mode Target Only. All enabled processors will operate in legacy mode only.
- 02h — Long Mode Target Only. All enabled processor will switch into long mode once.
- 03h — Mixed Mode Target. Processors may switch between legacy mode and long mode, or the preferred mode for system software is unknown. This value instructs the BIOS to use settings that are valid in all modes.
- All other values are reserved.

The Detect Target Operating Mode callback outputs are stored in the AH register and CF (carry flag in the EFLAGS register), and the values of other registers are not modified. The following output values are possible:

- AH = 00h and CF = 0, if the callback is implemented and the value in BL is supported.
- AH = 00h and CF = 1, if the callback is implemented and the value in BL is reserved. This indicates an error; the target operating mode is set to Legacy Mode Target Only.
- AH = 86h and CF = 1, if the callback is not supported.

## 12.22 SMM Issues

The processor includes support for system management mode (SMM). The functionality of the AMD Athlon™ 64 processor or AMD Opteron™ processor SMM is a superset of the Pentium® processor functionality.

- The AMD Athlon™ 64 and AMD Opteron™ Processors implement the SMM remapping and control registers as model-specific registers on the CPU.
- Implement the SMM state-save area in a manner compatible with the description of SMM found in Chapter 6, “System Management Mode (SMM).”

Program the model-specific registers to set the SMM memory base, local address, destination address, memory type, size and control, etc. See Chapter 6, “System Management Mode (SMM).”

## 13 Processor Configuration Registers

This chapter includes descriptions of two types of model-specific registers, as follows:

- General model-specific registers (see page 343)
- AMD Athlon™ 64 and AMD Opteron™ model-specific registers (see page 366)

### 13.1 General Model-Specific Registers

Table 81 is a listing of the general model-specific registers supported by the processor, presented in ascending hexadecimal address order. Register descriptions follow the table, organized according to the following functions:

- System software (see page 345)
- Memory typing (see page 347)
- APIC (see page 364)
- Machine check architecture (see page 192)
- Software debug (see page 365)
- Performance monitoring (see page 366)

**Table 81. General MSRs**

Address	Register Name	Description
0010h	TSC	page 366
001Bh	APIC_BASE	page 364
002Ah	EBL_CR_POWERON	page 364
00FEh	MTRRcap	page 347
0174h	SYSENTER_CS	page 345
0175h	SYSENTER_ESP	page 346
0176h	SYSENTER_EIP	page 346
0179h	MCG_CAP	page 193
017Ah	MCG_STATUS	page 193
017Bh	MCG_CTL	page 194

**Table 81. General MSRs (Continued)**

Address	Register Name	Description
01D9h	DebugCtl	page 365
01DBh	LastBranchFromIP	page 365
01DCh	LastBranchToIP	page 365
01DDh	LastExceptionFromIP	page 365
01DEh	LastExceptionToIP	page 365
0200–020Eh Even	MTRRphysBase[7:0]	page 348
0201–020Fh Odd	MTRRphysMask[7:0]	page 348
0250h	MTRRfix64K_00000	page 349
0258h	MTRRfix16K_80000	page 350
0259h	MTRRfix16K_A0000	page 351
0268h	MTRRfix4K_C0000	page 353
0269h	MTRRfix4K_C8000	page 354
026Ah	MTRRfix4K_D0000	page 355
026Bh	MTRRfix4K_D8000	page 356
026Ch	MTRRfix4K_E0000	page 357
026Dh	MTRRfix4K_E8000	page 358
026Eh	MTRRfix4K_F0000	page 360
026Fh	MTRRfix4K_F8000	page 361
0277h	PAT	page 362
02FFh	MTRRdefType	page 363
0400h	MC0_CTL	page 199
0401h	MC0_STATUS	page 200
0402h	MC0_ADDR	page 203
0403h,	MC0_MISC	not supported
0404h	MC1_CTL	page 204
0405h	MC1_STATUS	page 206
0406h	MC1_ADDR	page 206
0407h	MC1_MISC	not supported



**Table 81. General MSRs (Continued)**

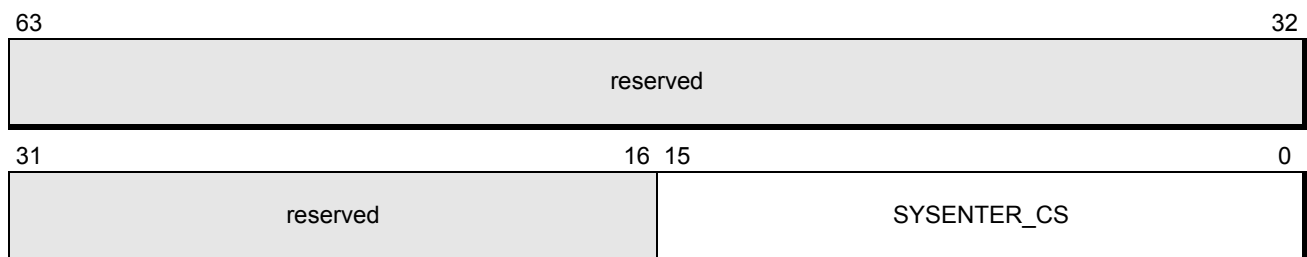
Address	Register Name	Description
0408h	MC2_CTL	page 204
0409h	MC2_STATUS	page 210
040Ah	MC2_ADDR	page 211
040Bh	MC2_MISC	not supported
040Ch	MC3_CTL	page 212
040Dh	MC3_STATUS	page 213
040Eh	MC3_ADDR	page 213
040Fh	MC3_MISC	not supported
0410h	MC4_CTL	page 213
0411h	MC4_STATUS	page 213
0412h	MC4_ADDR	page 213
0413h	MC4_MISC	not supported

## 13.1.1 System Software Registers

### 13.1.1.1 SYSENTER\_CS Register

This register contains the code segment selector used by the SYSENTER and SYSEXIT instructions. See the SYSENTER and SYSEXIT section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

#### SYSENTER\_CS Register

**MSR 0174h**


Bit	Mnemonic	Function	R/W	Reset
63–32	reserved	RAZ	R	0
31–16	reserved	SBZ	R/W	0
15–0	SYSENTER_CS	SYSENTER/SYSEXIT code segment selector	R/W	0

## Field Descriptions

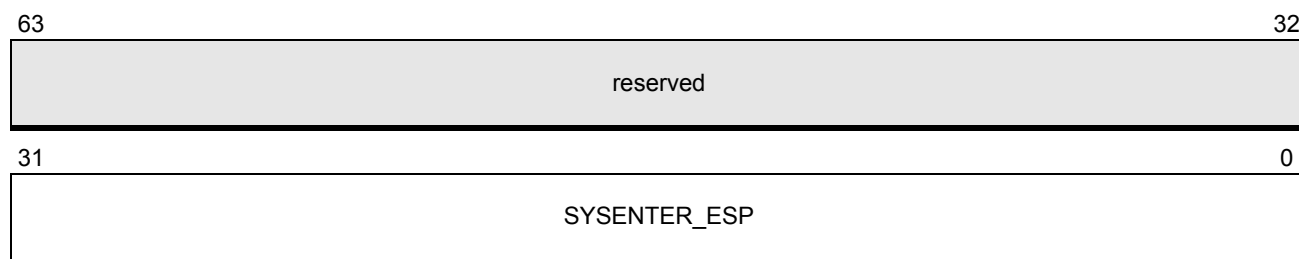
**SYSENTER/SYSEXIT Code Segment Selector (SYSENTER\_CS)—Bits 15–0.**

### 13.1.1.2 SYSENTER\_ESP Register

This register contains the stack pointer used by the SYSENTER and SYSEXIT instructions. See the SYSENTER and SYSEXIT section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

#### SYSENTER\_ESP Register

**MSR 0175h**



Bit	Mnemonic	Function	R/W	Reset
63–32	reserved	RAZ	R	0
31–0	SYSENTER_ESP	SYSENTER/SYSEXIT stack pointer	R/W	0

## Field Descriptions

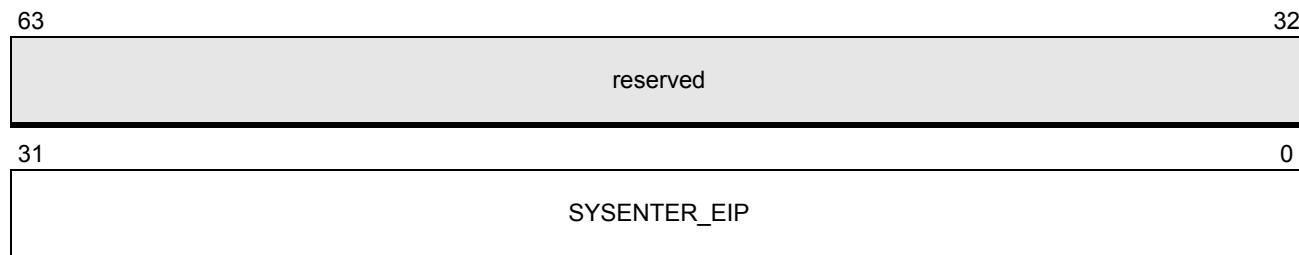
**SYSENTER/SYSEXIT Stack Pointer (SYSENTER\_ESP)—Bits 31–0.**

### 13.1.1.3 SYSENTER\_EIP Register

This register contains the instruction pointer used by the SYSENTER and SYSEXIT instructions. See the SYSENTER and SYSEXIT section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

#### SYSENTER\_EIP Register

**MSR 0176h**



Bit	Mnemonic	Function	R/W	Reset
63–32	reserved	RAZ	R	0
31–0	SYSENTER_EIP	SYSENTER/SYSEXIT instruction pointer	R/W	0

## Field Descriptions

**SYSENTER/SYSEXIT Instruction Pointer (SYSENTER\_EIP)**—Bits 31–0.

## 13.1.2 Memory Typing Registers

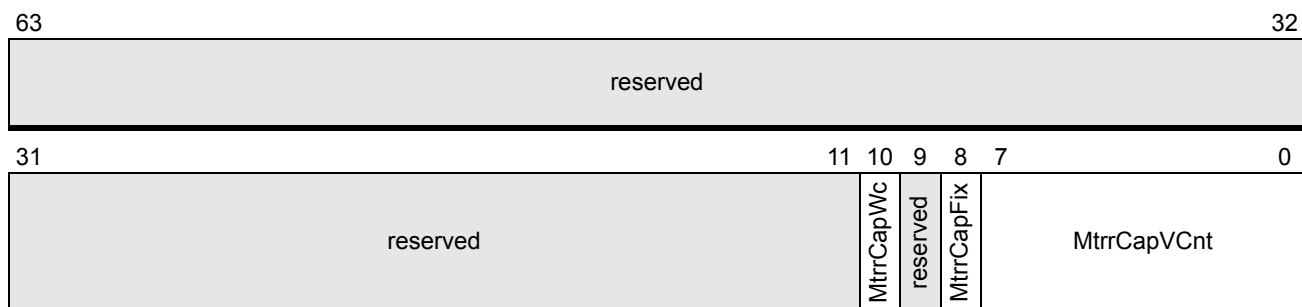
### 13.1.2.1 MTRRcap Register

This is a read-only register that returns information about the processors MTRR capabilities. See the “Using MTRRs” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

The MTRRcap register is a read-only status register. Attempting to modify this register will result in a #GP(0).

### MTRRcap Register

**MSR 00FEh**



Bit	Mnemonic	Function	R/W
63–11	reserved	RAZ	R
10	MtrrCapWc	Write-combining memory type	R
9	reserved	RAZ	R
8	MtrrCapFix	Fixed range register	R
7–0	MtrrCapVCnt	Variable range registers count	R

## Field Descriptions

**Variable Range Registers Count (MtrrCapVCnt)**—Bits 7–0. Indicates number of variable range registers.

**Fixed Range Registers (MtrrCapFix)**—Bit 8. Indicates fixed range register capability.

**Write-Combining Memory Type (MtrrCapWc)**—Bit 10. Indicates write-combining memory type capability.

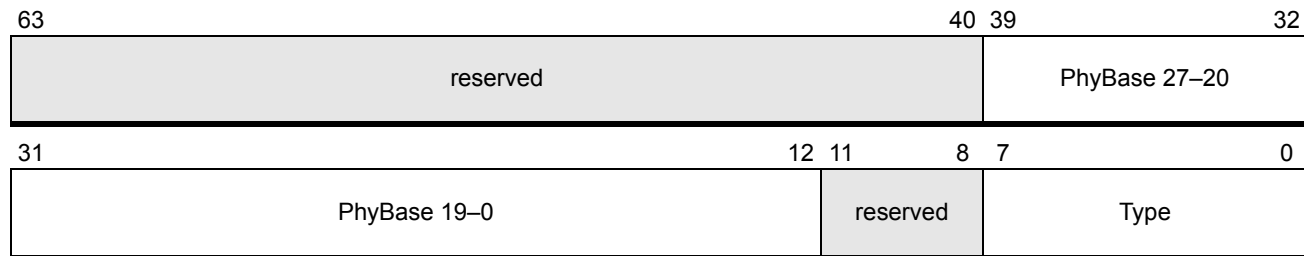
### 13.1.2.2 MTRRphysBase*i* Registers

These registers define the base address and memory type for each of the variable MTRRs. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Attempting to modify reserved bits in MTRRphysBase*i* will result in a #GP(0).

#### MTRRphysBase0–7 Registers

**MSRs 0200h, 0202h, 0204h, 0206h,  
0208h, 020Ah, 020Ch, 020Eh**



Bit	Mnemonic	Function	R/W	Reset
63–40	reserved	MBZ		0
39–12	PhyBase	Base address	R/W	U
11–8	reserved	MBZ		0
7–0	Type	Memory type	R/W	U

### Field Descriptions

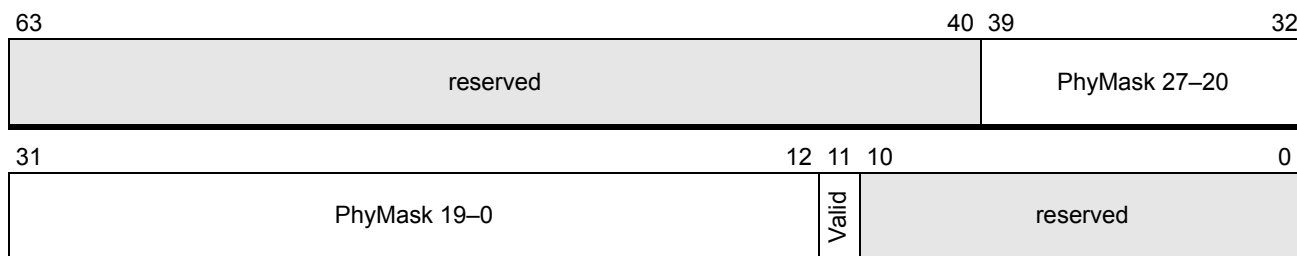
**Memory Type (Type)**—Bits 7–0. Specifies memory type for this memory range.

**Base Address (PhysBase)**—Bits 39–12. Specifies base address for this memory range

### 13.1.2.3 MTRRphysMask*i* Registers

These registers define the address mask and valid bit for each of the variable MTRRs. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Attempting to modify reserved bits in MTRRphysMask*i* will result in a #GP(0).

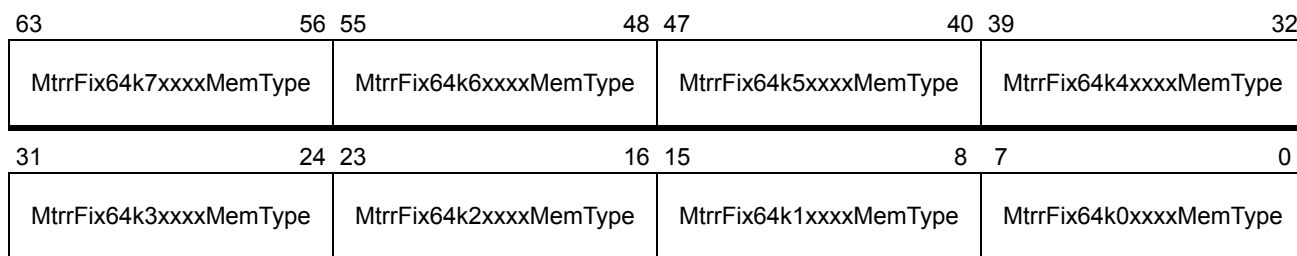
**MTRRphysMask0–7 Registers****MSRs 0201h, 0203h, 0205, 0207h,  
0209h, 020Bh, 020Dh, 020Fh**

Bit	Mnemonic	Function	R/W	Reset
63–40	reserved	MBZ		0
39–12	PhysMask	Address mask	R/W	U
11	Valid	MTRR is valid	R/W	0
10–0	reserved	MBZ		0

**Field Descriptions****MTRR is Valid (Valid)**—Bit 11. Indicated MTRR is valid.**Address Mask (PhysMask)**—Bits 39–12. Specifies the address mask for this memory range.**13.1.2.4 MTRRfix64K\_00000 Register**

This register controls the memory types for the first 512 Kbyte of physical memory. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix64K\_00000 Register****MSR 0250h**

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix64k7xxxxMemType	Memory type address 70000–7FFFFh	R/W	U
55–48	MtrrFix64k6xxxxMemType	Memory type address 60000–6FFFFh	R/W	U
47–40	MtrrFix64k5xxxxMemType	Memory type address 50000–5FFFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

Bit	Mnemonic	Function	R/W	Reset
39–32	MtrrFix64k4xxxxMemType	Memory type address 40000–4FFFFh	R/W	U
31–24	MtrrFix64k3xxxxMemType	Memory type address 30000–3FFFFh	R/W	U
23–16	MtrrFix64k2xxxxMemType	Memory type address 20000–2FFFFh	R/W	U
15–8	MtrrFix64k1xxxxMemType	Memory type address 10000–1FFFFh	R/W	U
7–0	MtrrFix64k0xxxxMemType	Memory type address 00000–0FFFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

## Field Descriptions

**Memory Type Address 00000–0FFFFh (MtrrFix64k0xxxxMemType)**—Bits 7–0. Memory type for physical address 00000–0FFFFh.

**Memory Type Address 10000–1FFFFh (MtrrFix64k1xxxxMemType)**—Bits 15–8. Memory type for physical address 10000–1FFFFh.

**Memory Type Address 20000–2FFFFh (MtrrFix64k2xxxxMemType)**—Bits 23–16. Memory type for physical address 20000–2FFFFh.

**Memory Type Address 30000–3FFFFh (MtrrFix64k3xxxxMemType)**—Bits 31–24. Memory type for physical address 30000–3FFFFh.

**Memory Type Address 40000–4FFFFh (MtrrFix64k4xxxxMemType)**—Bits 39–32. Memory type for physical address 40000–4FFFFh.

**Memory Type Address 50000–5FFFFh (MtrrFix64k5xxxxMemType)**—Bits 47–40. Memory type for physical address 50000–5FFFFh.

**Memory Type Address 60000–6FFFFh (MtrrFix64k6xxxxMemType)**—Bits 55–48. Memory type for physical address 60000–6FFFFh.

**Memory Type Address 70000–7FFFFh (MtrrFix64k7xxxxMemType)**—Bits 63–56. Memory type for physical address 70000–7FFFFh.

### 13.1.2.5 MTRRfix16K\_80000 Register

This register controls the memory types for physical memory addresses 80000–9FFFFh. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

#### MTRRfix16K\_80000 Register

**MSR 0258h**

63	56	55	48	47	40	39	32
MtrrFix16k9CxxxMemType	MtrrFix16k98xxxMemType	MtrrFix16k94xxxMemType	MtrrFix16k90xxxMemType				

31	24	23	16	15	8	7	0	
MtrrFix16k8CxxxMemType				MtrrFix16k88xxxMemType		MtrrFix16k84xxxMemType		MtrrFix16k80xxxMemType

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix16k9CxxxMemType	Memory type address 9C000–9FFFFh	R/W	U
55–48	MtrrFix16k98xxxMemType	Memory type address 98000–9BFFFh	R/W	U
47–40	MtrrFix16k94xxxMemType	Memory type address 94000–97FFFh	R/W	U
39–32	MtrrFix16k90xxxMemType	Memory type address 90000–93FFFh	R/W	U
31–24	MtrrFix16k8CxxxMemType	Memory type address 8C000–8FFFFh	R/W	U
23–16	MtrrFix16k88xxxMemType	Memory type address 88000–8BFFFh	R/W	U
15–8	MtrrFix16k84xxxMemType	Memory type address 84000–87FFFh	R/W	U
7–0	MtrrFix16k80xxxMemType	Memory type address 80000–83FFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

## Field Descriptions

**Memory Type Address 80000–83FFFh (MtrrFix16k80xxxMemType)**—Bits 7–0. Memory type for physical address 80000–83FFFh.

**Memory Type Address 84000–87FFFh (MtrrFix16k84xxxMemType)**—Bits 15–8. Memory type for physical address 84000–87FFFh.

**Memory Type Address 88000–8BFFFh (MtrrFix16k88xxxMemType)**—Bits 23–16. Memory type for physical address 88000–8BFFFh.

**Memory Type Address 8C000–8FFFFh (MtrrFix16k8CxxxMemType)**—Bits 31–24. Memory type for physical address 8C000–8FFFFh.

**Memory Type Address 90000–93FFFh (MtrrFix16k90xxxMemType)**—Bits 39–32. Memory type for physical address 90000–93FFFh.

**Memory Type Address 94000–97FFFh (MtrrFix16k94xxxMemType)**—Bits 47–40. Memory type for physical address 94000–97FFFh.

**Memory Type Address 98000–9BFFFh (MtrrFix16k98xxxMemType)**—Bits 55–48. Memory type for physical address 98000–9BFFFh.

**Memory Type Address 9C000–9FFFFh (MtrrFix16k9CxxxMemType)**—Bits 63–56. Memory type for physical address 9C000–9FFFFh.

### 13.1.2.6 MTRRfix16K\_A0000 Register

This register controls the memory types for physical memory addresses A0000–BFFFFh. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix16K\_A0000 Register**
**MSR 0259h**

63	56	55	48	47	40	39	32
MtrrFix16kBCxxxMemType	MtrrFix16kB8xxxMemType	MtrrFix16kB4xxxMemType	MtrrFix16kB0xxxMemType				
31	24	23	16	15	8	7	0
MtrrFix16kACxxxMemType	MtrrFix16kA8xxxMemType	MtrrFix16kA4xxxMemType	MtrrFix16kA0xxxMemType				

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix16kBCxxxMemType	Memory type address BC000–BFFFFh	R/W	U
55–48	MtrrFix16kB8xxxMemType	Memory type address B8000–BBFFFh	R/W	U
47–40	MtrrFix16kB4xxxMemType	Memory type address B4000–B7FFFh	R/W	U
39–32	MtrrFix16kB0xxxMemType	Memory type address B0000–B3FFFh	R/W	U
31–24	MtrrFix16kACxxxMemType	Memory type address AC000–AFFFFh	R/W	U
23–16	MtrrFix16kA8xxxMemType	Memory type address A8000–ABFFFh	R/W	U
15–8	MtrrFix16kA4xxxMemType	Memory type address A4000–A7FFFh	R/W	U
7–0	MtrrFix16kA0xxxMemType	Memory type address A0000–A3FFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

**Field Descriptions**

**Memory Type Address A0000–A3FFFh (MtrrFix16kA0xxxMemType)**—Bits 7–0. Memory type for physical address A0000–A3FFFh.

**Memory Type Address A4000–A7FFFh (MtrrFix16kA4xxxMemType)**—Bits 15–8. Memory type for physical address A4000–A7FFFh.

**Memory Type Address A8000–ABFFFh (MtrrFix16kA8xxxMemType)**—Bits 23–16. Memory type for physical address A8000–ABFFFh.

**Memory Type Address AC000–AFFFFh (MtrrFix16kACxxxMemType)**—Bits 31–24. Memory type for physical address AC000–AFFFFh.

**Memory Type Address B0000–B3FFFh (MtrrFix16kB0xxxMemType)**—Bits 39–32. Memory type for physical address B0000–B3FFFh.

**Memory Type Address B4000–B7FFFh (MtrrFix16kB4xxxMemType)**—Bits 47–40. Memory type for physical address B4000–B7FFFh.

**Memory Type Address B8000–BBFFFh (MtrrFix16kB8xxxMemType)**—Bits 55–48. Memory type for physical address B8000–BBFFFh.

**Memory Type Address BC000–BFFFFh (MtrrFix16kBCxxxMemType)**—Bits 63–56. Memory type for physical address BC000–BFFFFh.



**13.1.2.7 MTRRfix4K\_C0000 Register**

This register controls the memory types for physical memory addresses C0000–C7FFFh. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix4K\_C0000 Register****MSR 0268h**

63	56	55	48	47	40	39	32
MtrrFix4kC7xxxMemType				MtrrFix4kC6xxxMemType			
MtrrFix4kC5xxxMemType				MtrrFix4kC4xxxMemType			
31	24	23	16	15	8	7	0
MtrrFix4kC3xxxMemType				MtrrFix4kC2xxxMemType			
MtrrFix4kC1xxxMemType				MtrrFix4kC0xxxMemType			

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix4kC7xxxMemType	Memory type address C7000–C7FFFh	R/W	U
55–48	MtrrFix4kC6xxxMemType	Memory type address C6000–C6FFFh	R/W	U
47–40	MtrrFix4kC5xxxMemType	Memory type address C5000–C5FFFh	R/W	U
39–32	MtrrFix4kC4xxxMemType	Memory type address C4000–C4FFFh	R/W	U
31–24	MtrrFix4kC3xxxMemType	Memory type address C3000–C3FFFh	R/W	U
23–16	MtrrFix4kC2xxxMemType	Memory type address C2000–C2FFFh	R/W	U
15–8	MtrrFix4kC1xxxMemType	Memory type address C1000–C1FFFh	R/W	U
7–0	MtrrFix4kC0xxxMemType	Memory type address C0000–C0FFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

**Field Descriptions**

**Memory Type Address C0000–C0FFFh (MtrrFix4kC0xxxMemType)**—Bits 7–0. Memory type for physical address C0000–C0FFFh.

**Memory Type Address C1000–C1FFFh (MtrrFix4kC1xxxMemType)**—Bits 15–8. Memory type for physical address C1000–C1FFFh.

**Memory Type Address C2000–C2FFFh (MtrrFix4kC2xxxMemType)**—Bits 23–16. Memory type for physical address C2000–C2FFFh.

**Memory Type Address C3000–C3FFFh (MtrrFix4kC3xxxMemType)**—Bits 31–24. Memory type for physical address C3000–C3FFFh.

**Memory Type Address C4000–C4FFFh (MtrrFix4kC4xxxMemType)**—Bits 39–32. Memory type for physical address C4000–C4FFFh.

**Memory Type Address C5000–C5FFFh (MtrrFix4kC5xxxMemType)**—Bits 47–40. Memory type for physical address C5000–C5FFFh.

**Memory Type Address C6000–C6FFFh (MtrrFix4kC6xxxMemType)**—Bits 55–48. Memory type for physical address C6000–C6FFFh.

**Memory Type Address C7000–C7FFFh (MtrrFix4kC7xxxMemType)**—Bits 63–56. Memory type for physical address C7000–C7FFFh.

### 13.1.2.8 MTRRfix4K\_C8000 Register

This register controls the memory types for physical memory addresses C8000–CFFFFh. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

#### MTRRfix4K\_C8000 Register

**MSR 0269h**

63	56	55	48	47	40	39	32
MtrrFix4kCFxxxMemType	MtrrFix4kCExxxMemType	MtrrFix4kCDxxxMemType	MtrrFix4kCCxxxMemType				
31	24	23	16	15	8	7	0
MtrrFix4kCBxxxMemType	MtrrFix4kCAxxxMemType	MtrrFix4kC9xxxMemType	MtrrFix4kC8xxxMemType				

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix4kCFxxxMemType	Memory type for address CF000–CFFFFh	R/W	U
55–48	MtrrFix4kCExxxMemType	Memory type for address CE000–CEFFFh	R/W	U
47–40	MtrrFix4kCDxxxMemType	Memory type for address CD000–CDFFFh	R/W	U
39–32	MtrrFix4kCCxxxMemType	Memory type for address CC000–CCFFFh	R/W	U
31–24	MtrrFix4kCBxxxMemType	Memory type for address CB000–CBFFFh	R/W	U
23–16	MtrrFix4kCAxxxMemType	Memory type for address CA000–CAFFFh	R/W	U
15–8	MtrrFix4kC9xxxMemType	Memory type for address C9000–C9FFFh	R/W	U
7–0	MtrrFix4kC8xxxMemType	Memory type for address C8000–C8FFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

### Field Descriptions

**Memory Type Address C8000–C8FFFh (MtrrFix4kC8xxxMemType)**—Bits 7–0. Memory type for physical address C8000–C8FFFh.

**Memory Type Address C9000–C9FFFh (MtrrFix4kC9xxxMemType)**—Bits 15–8. Memory type for physical address C9000–C9FFFh.

**Memory Type Address CA000–CAFFFh (MtrrFix4kCAxxxMemType)**—Bits 23–16. Memory type for physical address CA000–CAFFFh.

**Memory Type Address CB000–CBFFFh (MtrrFix4kCBxxxMemType)**—Bits 31–24. Memory type for physical address CB000–CBFFFh.

**Memory Type Address CC000–CCFFFh (MtrrFix4kCCxxxMemType)**—Bits 39–32. Memory type for physical address CC000–CCFFFh.

**Memory Type Address CD000–CDFFFh (MtrrFix4kCDxxxMemType)**—Bits 47–40. Memory type for physical address CD000–CDFFFh.

**Memory Type Address CE000–CEFFFh (MtrrFix4kCExxxMemType)**—Bits 55–48. Memory type for physical address CE000–CEFFFh.

**Memory Type Address CF000–CFFFh (MtrrFix4kCFxxxMemType)**—Bits 63–56. Memory type for physical address CF000–CFFFh.

### 13.1.2.9 MTRRfix4K\_D0000 Register

This register controls the memory types for physical memory addresses D0000–D7FFF. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

#### MTRRfix4K\_D0000 Register

**MSR 026Ah**

63	56	55	48	47	40	39	32
MtrrFix4kD7xxxMemType	MtrrFix4kD6xxxMemType	MtrrFix4kD5xxxMemType	MtrrFix4kD4xxxMemType				
31	24	23	16	15	8	7	0
MtrrFix4kD3xxxMemType	MtrrFix4kD2xxxMemType	MtrrFix4kD1xxxMemType	MtrrFix4kD0xxxMemType				

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix4kD7xxxMemType	Memory type address D7000–D7FFFh	R/W	U
55–48	MtrrFix4kD6xxxMemType	Memory type address D6000–D6FFFh	R/W	U
47–40	MtrrFix4kD5xxxMemType	Memory type address D5000–D5FFFh	R/W	U
39–32	MtrrFix4kD4xxxMemType	Memory type address D4000–D4FFFh	R/W	U
31–24	MtrrFix4kD3xxxMemType	Memory type address D3000–D3FFFh	R/W	U
23–16	MtrrFix4kD2xxxMemType	Memory type address D2000–D2FFFh	R/W	U
15–8	MtrrFix4kD1xxxMemType	Memory type address D1000–D1FFFh	R/W	U
7–0	MtrrFix4kD0xxxMemType	Memory type address D0000–D0FFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

## Field Descriptions

**Memory Type Address D0000–D0FFFh (MtrrFix4kD0xxxMemType)**—Bits 7–0. Memory type for physical address D0000–D0FFFh.

**Memory Type Address D1000–D1FFFh (MtrrFix4kD1xxxMemType)**—Bits 15–8. Memory type for physical address D1000–D1FFFh.

**Memory Type Address D2000–D2FFFh (MtrrFix4kD2xxxMemType)**—Bits 23–16. Memory type for physical address D2000–D2FFFh.

**Memory Type Address D3000–D3FFFh (MtrrFix4kD3xxxMemType)**—Bits 31–24. Memory type for physical address D3000–D3FFFh.

**Memory Type Address D4000–D4FFFh (MtrrFix4kD4xxxMemType)**—Bits 39–32. Memory type for physical address D4000–D4FFFh.

**Memory Type Address D5000–D5FFFh (MtrrFix4kD5xxxMemType)**—Bits 47–40. Memory type for physical address D5000–D5FFFh.

**Memory Type Address D6000–D6FFFh (MtrrFix4kD6xxxMemType)**—Bits 55–48. Memory type for physical address D6000–D6FFFh.

**Memory Type Address D7000–D7FFFh (MtrrFix4kD7xxxMemType)**—Bits 63–56. Memory type for physical address D7000–D7FFFh.

### 13.1.2.10 MTRRfix4K\_D8000 Register

This register controls the memory types for physical memory addresses D8000–DFFFFh. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

#### MTRRfix4K\_D8000 Register

**MSR 026Bh**

63	56	55	48	47	40	39	32
MtrrFix4kDFxxxMemType	MtrrFix4kDExxxMemType	MtrrFix4kDDxxxMemType	MtrrFix4kDCxxxMemType				
31	24	23	16	15	8	7	0
MtrrFix4kDBxxxMemType	MtrrFix4kDAxxxMemType	MtrrFix4kD9xxxMemType	MtrrFix4kD8xxxMemType				

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix4kDFxxxMemType	Memory type address DF000–DFFFFh	R/W	U
55–48	MtrrFix4kDExxxMemType	Memory type address DE000–DEFFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

Bit	Mnemonic	Function	R/W	Reset
47–40	MtrrFix4kDDxxxMemType	Memory type address DD000–DDFFFh	R/W	U
39–32	MtrrFix4kDCxxxMemType	Memory type address DC000–DCFFFh	R/W	U
31–24	MtrrFix4kDBxxxMemType	Memory type address DB000–DBFFFh	R/W	U
23–16	MtrrFix4kDAxxxMemType	Memory type address DA000–DAFFFh	R/W	U
15–8	MtrrFix4kD9xxxMemType	Memory type address D9000–D9FFFh	R/W	U
7–0	MtrrFix4kD8xxxMemType	Memory type address D8000–D8FFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

## Field Descriptions

**Memory Type Address D8000–D8FFFh (MtrrFix4kD8xxxMemType)**—Bits 7–0. Memory type for physical address D8000–D8FFFh.

**Memory Type Address D9000–D9FFFh (MtrrFix4kD9xxxMemType)**—Bits 15–8. Memory type for physical address D9000–D9FFFh.

**Memory Type Address DA000–DAFFFh (MtrrFix4kDAxxxMemType)**—Bits 23–16. Memory type for physical address DA000–DAFFFh.

**Memory Type Address DB000–DBFFFh (MtrrFix4kDBxxxMemType)**—Bits 31–24. Memory type for physical address DB000–DBFFFh.

**Memory Type Address DC000–DCFFFh (MtrrFix4kDCxxxMemType)**—Bits 39–32. Memory type for physical address DC000–DCFFFh.

**Memory Type Address DD000–DDFFFh (MtrrFix4kDDxxxMemType)**—Bits 47–40. Memory type for physical address DD000–DDFFFh.

**Memory Type Address DE000–DEFFFh (MtrrFix4kDExxxMemType)**—Bits 55–48. Memory type for physical address DE000–DEFFFh.

**Memory Type Address DF000–DFFFFh (MtrrFix4kDFxxxMemType)**—Bits 63–56. Memory type for physical address DF000–DFFFFh.

### 13.1.2.11 MTRRfix4K\_E0000 Register

This register controls the memory types for physical memory addresses E0000–E7FFFh. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

#### MTRRfix4K\_E0000 Register

**MSR 026Ch**

63	56 55	48 47	40 39	32
MtrrFix4kE7xxxMemType	MtrrFix4kE6xxxMemType	MtrrFix4kE5xxxMemType	MtrrFix4kE4xxxMemType	

31	24	23	16	15	8	7	0	
MtrrFix4kE3xxxMemType				MtrrFix4kE2xxxMemType		MtrrFix4kE1xxxMemType		MtrrFix4kE0xxxMemType

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix4kE7xxxMemType	Memory type for physical addr E7000–E7FFFh	R/W	U
55–48	MtrrFix4kE6xxxMemType	Memory type for physical addr E6000–E6FFFh	R/W	U
47–40	MtrrFix4kE5xxxMemType	Memory type for physical addr E5000–E5FFFh	R/W	U
39–32	MtrrFix4kE4xxxMemType	Memory type for physical addr E4000–E4FFFh	R/W	U
31–24	MtrrFix4kE3xxxMemType	Memory type for physical addr E3000–E3FFFh	R/W	U
23–16	MtrrFix4kE2xxxMemType	Memory type for physical addr E2000–E2FFFh	R/W	U
15–8	MtrrFix4kE1xxxMemType	Memory type for physical addr E1000–E1FFFh	R/W	U
7–0	MtrrFix4kE0xxxMemType	Memory type for physical addr E0000–E0FFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

## Field Descriptions

**Memory Type Address E0000–E0FFFh (MtrrFix4kE0xxxMemType)**—Bits 7–0. Memory type for physical address E0000–E0FFFh.

**Memory Type Address E1000–E1FFFh (MtrrFix4kE1xxxMemType)**—Bits 15–8. Memory type for physical address E1000–E1FFFh.

**Memory Type Address E2000–E2FFFh (MtrrFix4kE2xxxMemType)**—Bits 23–16. Memory type for physical address E2000–E2FFFh.

**Memory Type Address E3000–E3FFFh (MtrrFix4kE3xxxMemType)**—Bits 31–24. Memory type for physical address E3000–E3FFFh.

**Memory Type Address E4000–E4FFFh (MtrrFix4kE4xxxMemType)**—Bits 39–32. Memory type for physical address E4000–E4FFFh.

**Memory Type Address E5000–E5FFFh (MtrrFix4kE5xxxMemType)**—Bits 47–40. Memory type for physical address E5000–E5FFFh.

**Memory Type Address E6000–E6FFFh (MtrrFix4kE6xxxMemType)**—Bits 55–48. Memory type for physical address E6000–E6FFFh.

**Memory Type Address E7000–E7FFFh (MtrrFix4kE7xxxMemType)**—Bits 63–56. Memory type for physical address E7000–E7FFFh.

### 13.1.2.12 MTRRfix4K\_E8000 Register

This register controls the memory types for physical memory addresses E8000–EFFFFh. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix4K\_E8000 Register****MSR 026Dh**

63	56 55	48 47	40 39	32
MtrrFix4kEFxxxMemType	MtrrFix4kEExxxMemType	MtrrFix4kEDxxxMemType	MtrrFix4kECxxxMemType	
31	24 23	16 15	8 7	0
MtrrFix4kEBxxxMemType	MtrrFix4kEAxxxMemType	MtrrFix4kE9xxxMemType	MtrrFix4kE8xxxMemType	

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix4kEFxxxMemType	Memory type for physical addr EF000–EFFFFh	R/W	U
55–48	MtrrFix4kEExxxMemType	Memory type for physical addr EE000–EFFFFh	R/W	U
47–40	MtrrFix4kEDxxxMemType	Memory type for physical addr ED000–EDFFFh	R/W	U
39–32	MtrrFix4kECxxxMemType	Memory type for physical addr EC000–ECFFFh	R/W	U
31–24	MtrrFix4kEBxxxMemType	Memory type for physical addr EB000–EBFFFh	R/W	U
23–16	MtrrFix4kEAxxxMemType	Memory type for physical addr EA000–EAFFFh	R/W	U
15–8	MtrrFix4kE9xxxMemType	Memory type for physical addr E9000–E9FFFh	R/W	U
7–0	MtrrFix4kE8xxxMemType	Memory type for physical addr E8000–E8FFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

**Field Descriptions**

**Memory Type Address E8000–E8FFFh (MtrrFix4kE8xxxMemType)**—Bits 7–0. Memory type for physical address E8000–E8FFFh.

**Memory Type Address E9000–E9FFFh (MtrrFix4kE9xxxMemType)**—Bits 15–8. Memory type for physical address E9000–E9FFFh.

**Memory Type Address EA000–EAFFFh (MtrrFix4kEAxxxMemType)**—Bits 23–16. Memory type for physical address EA000–EAFFFh.

**Memory Type Address EB000–EBFFFh (MtrrFix4kEBxxxMemType)**—Bits 31–24. Memory type for physical address EB000–EBFFFh.

**Memory Type Address EC000–ECFFFh (MtrrFix4kECxxxMemType)**—Bits 39–32. Memory type for physical address EC000–ECFFFh.

**Memory Type Address ED000–EDFFFh (MtrrFix4kEDxxxMemType)**—Bits 47–40. Memory type for physical address ED000–EDFFFh.

**Memory Type Address EE000–EFFFFh (MtrrFix4kEExxxMemType)**—Bits 55–48. Memory type for physical address EE000–EFFFFh.

**Memory Type Address EF000–EFFFFh (MtrrFix4kEFxxxMemType)**—Bits 63–56. Memory type for physical address EF000–EFFFFh.

**13.1.2.13 MTRRfix4K\_F0000 Register**

This register controls the memory types for physical memory addresses F0000–F7FFFh. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

**MTRRfix4K\_F0000 Register****MSR 026Eh**

63	56	55	48	47	40	39	32
MtrrFix4kF7xxxMemType				MtrrFix4kF6xxxMemType			
MtrrFix4kF5xxxMemType				MtrrFix4kF4xxxMemType			
31	24	23	16	15	8	7	0
MtrrFix4kF3xxxMemType				MtrrFix4kF2xxxMemType			
MtrrFix4kF1xxxMemType				MtrrFix4kF0xxxMemType			

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix4kF7xxxMemType	Memory type address F7000–F7FFFh	R/W	U
55–48	MtrrFix4kF6xxxMemType	Memory type address F6000–F6FFFh	R/W	U
47–40	MtrrFix4kF5xxxMemType	Memory type address F5000–F5FFFh	R/W	U
39–32	MtrrFix4kF4xxxMemType	Memory type address F4000–F4FFFh	R/W	U
31–24	MtrrFix4kF3xxxMemType	Memory type address F3000–F3FFFh	R/W	U
23–16	MtrrFix4kF2xxxMemType	Memory type address F2000–F2FFFh	R/W	U
15–8	MtrrFix4kF1xxxMemType	Memory type address F1000–F1FFFh	R/W	U
7–0	MtrrFix4kF0xxxMemType	Memory type address F0000–F0FFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

**Field Descriptions**

**Memory Type Address F0000–F0FFFh (MtrrFix4kF0xxxMemType)**—Bits 7–0. Memory type for physical address F0000–F0FFFh.

**Memory Type Address F1000–F1FFFh (MtrrFix4kF1xxxMemType)**—Bits 15–8. Memory type for physical address F1000–F1FFFh.

**Memory Type Address F2000–F2FFFh (MtrrFix4kF2xxxMemType)**—Bits 23–16. Memory type for physical address F2000–F2FFFh.

**Memory Type Address F3000–F3FFFh (MtrrFix4kF3xxxMemType)**—Bits 31–24. Memory type for physical address F3000–F3FFFh.

**Memory Type Address F4000–F4FFFh (MtrrFix4kF4xxxMemType)**—Bits 39–32. Memory type for physical address F4000–F4FFFh.



**Memory Type Address F5000–F5FFFh (MtrrFix4kF5xxxMemType)**—Bits 47–40. Memory type for physical address F5000–F5FFFh.

**Memory Type Address F6000–F6FFFh (MtrrFix4kF6xxxMemType)**—Bits 55–48. Memory type for physical address F6000–F6FFFh.

**Memory Type Address F7000–F7FFFh (MtrrFix4kF7xxxMemType)**—Bits 63–56. Memory type for physical address F7000–F7FFFh.

#### 13.1.2.14 MTRRfix4K\_F8000 Register

This register controls the memory types for physical memory addresses F8000–FFFFFFh. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0).

#### MTRRfix4K\_F8000 Register

**MSR 026Fh**

63	56	55	48	47	40	39	32
MtrrFix4kFFxxxMemType	MtrrFix4kFExxxMemType	MtrrFix4kFDxxxMemType	MtrrFix4kFCxxxMemType				
31	24	23	16	15	8	7	0
MtrrFix4kFBxxxMemType	MtrrFix4kFAxxxMemType	MtrrFix4kF9xxxMemType	MtrrFix4kF8xxxMemType				

Bit	Mnemonic	Function	R/W	Reset
63–56	MtrrFix4kFFxxxMemType	Memory type address FF000–FFFFFFh	R/W	U
55–48	MtrrFix4kFExxxMemType	Memory type address FE000–FEFFFh	R/W	U
47–40	MtrrFix4kFDxxxMemType	Memory type address FD000–FDFFFh	R/W	U
39–32	MtrrFix4kFCxxxMemType	Memory type address FC000–FCFFFh	R/W	U
31–24	MtrrFix4kFBxxxMemType	Memory type address FB000–FBFFFh	R/W	U
23–16	MtrrFix4kFAxxxMemType	Memory type address FA000–FAFFFh	R/W	U
15–8	MtrrFix4kF9xxxMemType	Memory type address F9000–F9FFFh	R/W	U
7–0	MtrrFix4kF8xxxMemType	Memory type address F8000–F8FFFh	R/W	U

\* Memory types must be set to values consistent with system hardware.

#### Field Descriptions

**Memory Type Address F8000–F8FFFh (MtrrFix4kF8xxxMemType)**—Bits 7–0. Memory type for physical address F8000–F8FFFh.

**Memory Type Address F9000–F9FFFh (MtrrFix4kF9xxxMemType)**—Bits 15–8. Memory type for physical address F9000–F9FFFh.

**Memory Type Address FA000–FAFFFh (MtrrFix4kFAxxxMemType)**—Bits 23–16. Memory type for physical address FA000–FAFFFh.

**Memory Type Address FB000–FBFFFh (MtrrFix4kFBxxxMemType)**—Bits 31–24. Memory type for physical address FB000–FBFFFh.

**Memory Type Address FC000–FCFFFh (MtrrFix4kFCxxxMemType)**—Bits 39–32. Memory type for physical address FC000–FCFFFh.

**Memory Type Address FD000–FDFFFh (MtrrFix4kFDxxxMemType)**—Bits 47–40. Memory type for physical address FD000–FDFFFh.

**Memory Type Address FE000–FEFFFh (MtrrFix4kFExxxMemType)**—Bits 55–48. Memory type for physical address FE000–FEFFFh.

**Memory Type Address FF000–FFFFFh (MtrrFix4kFFxxxMemType)**—Bits 63–56. Memory type for physical address FF000–FFFFFh.

### 13.1.2.15 PAT Register

This register contains the eight page attribute fields used for specifying memory types for pages. See the “Page-Attribute Table Mechanism” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting any range to an undefined memory type will result in a #GP(0). Setting any reserved bits will result in a #GP(0).

#### PAT Register

**MSR 0277h**

63	59	58	56	55	51	50	48	47	43	42	40	39	35	34	32
reserved	PA7			reserved	PA6			reserved	PA5			reserved	PA4		
31	27	26	24	23	19	18	16	15	11	10	8	7	3	2	0
reserved	PA3			reserved	PA2			reserved	PA1			reserved	PA0		

Bit	Mnemonic	Function	R/W	Reset
63–59	reserved	MBZ		0
58–56	PA7	Memory type for Page Attribute index 7	R/W	0
55–51	reserved	MBZ		0
50–48	PA6	Memory type for Page Attribute index 6	R/W	7
47–43	reserved	MBZ		0
42–40	PA5	Memory type for Page Attribute index 5	R/W	4
39–35	reserved	MBZ		0
34–32	PA4	Memory type for Page Attribute index 4	R/W	6
31–27	reserved	MBZ		0

Bit	Mnemonic	Function	R/W	Reset
26–24	PA3	Memory type for Page Attribute index 3	R/W	0
23–19	reserved	MBZ		0
18–16	PA2	Memory type for Page Attribute index 2	R/W	7
15–11	reserved	MBZ		0
10–8	PA1	Memory type for Page Attribute index 1	R/W	4
7–3	reserved	MBZ		0
2–0	PA0	Memory type for Page Attribute index 0	R/W	6

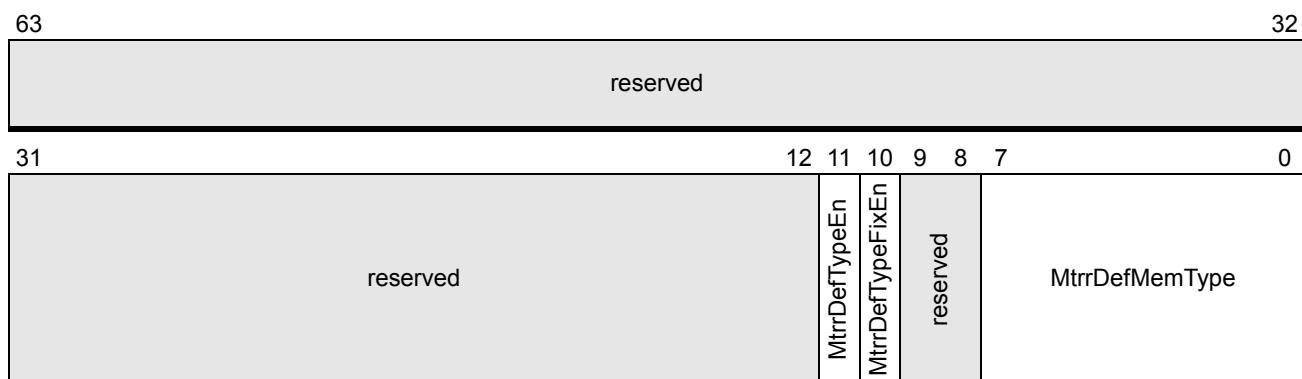
### 13.1.2.16 MTRRdefType Register

This register enables the MTRRs and defines the default memory type for memory not within one of the MTRR ranges. See the “Memory Type Range Registers” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

Setting MtrrDefMemType to an undefined memory type will result in a #GP(0). Setting any reserved bits will result in a #GP(0).

#### MTRRdefType Register

MSR 02FFh



Bit	Mnemonic	Function	R/W	Reset
63–12	reserved	MBZ		0
11	MtrrDefTypeEn	Enable MTRRs	R/W	0
10	MtrrDefTypeFixEn	Enable Fixed Range MTRRs	R/W	0
9–8	reserved			0
7–0	MtrrDefMemType	Default Memory Type	R/W	0

\* Memory types must be set to values consistent with system hardware.

### Field Descriptions

**Default Memory Type (MtrrDefMemType)**—Bit 7–0.

**Enable Fixed Range MTRRs (MtrrDefTypeFixEn)**—Bit 10.

**Enable MTRRs (MtrrDefTypeEn)—Bit 11.**

### 13.1.3 APIC Registers

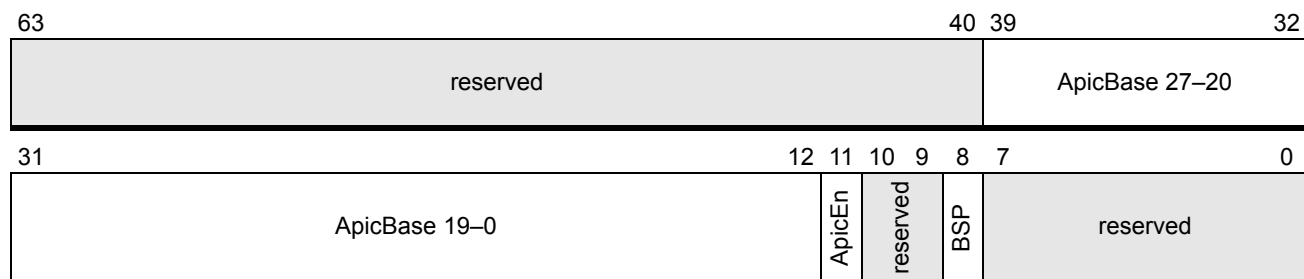
#### 13.1.3.1 APIC\_BASE Register

This register enables APIC and defines the APIC base address. It also identifies the Boot Strap Processor (BSP).

Setting any reserved bits will result in a #GP(0).

#### APIC\_BASE Register

**MSR 001Bh**



Bit	Mnemonic	Function	R/W	Reset
63–40	reserved	MBZ		0
39–12	ApicBase	APIC Base Address	R/W	00FEE00h
11	ApicEn	Enable APIC	R/W	0
10–9	reserved	MBZ		0
8	BSP	Boot Strap Processor	R/W	1 if uniprocessor or bsp of multiprocessor system
7–0	reserved	MBZ		0

\* APIC configuration must be consistent with system hardware.

#### Field Descriptions

**Boot Strap Processor (Bsp)—Bit 8.**

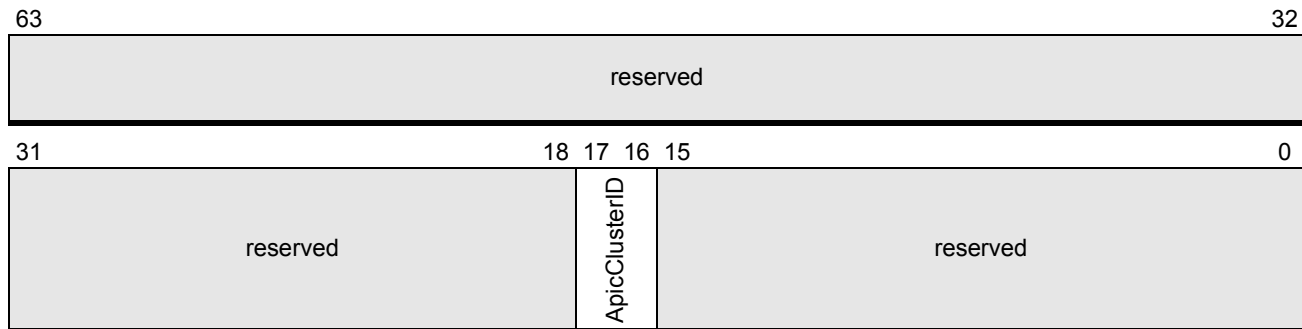
**Enable APIC (ApicEn)—Bit 11.**

**APIC Base Address (ApicBase)—Bits 39–12.**

#### 13.1.3.2 EBL\_CR\_POWERON Register

This read-only register contains the APIC cluster ID.

Attempting to write to this register will result in a #GP(0).

**EBL\_CR\_POWERON Register****MSR 002Ah**

Bit	Mnemonic	Function	R/W
63–18	reserved	MBZ	
17–16	ApicClusterID	APIC Cluster ID	R
15–0	reserved	MBZ	

**Field Descriptions**

**APIC Cluster ID (ApicClusterID)**—Bits 17–16.

**13.1.4 Software Debug Registers**

The AMD Athlon™ 64 and AMD Opteron™ Processors incorporate extensive debug features. These include the following control, status, and control-transfer recording MSRs.

- **DebugCtl Register (MSR 01D9h)**—Provides additional debug controls over control-transfer recording and single stepping, as well as external-breakpoint reporting and trace messages.
- **LastBranchFromIP Register (MSR 01DBh)**—Loaded with the segment offset of the branch instruction.
- **LastBranchToIP Register (MSR 01DCh)**—Holds the target rIP of the last branch that occurred before an exception or interrupt.
- **LastExceptionFromIP Register (MSR 01DDh)**—Holds the source rIP of the last branch that occurred before the exception or interrupt.
- **LastExceptionToIP Register (MSR 01DEh)**—Holds the target rIP of the last branch that occurred before the exception or interrupt.

For more detailed information on the use of these MSRs, see the “Debug and Performance Resources” section in Volume 2 of the *AMD64 Architecture Programmer's Manual*.

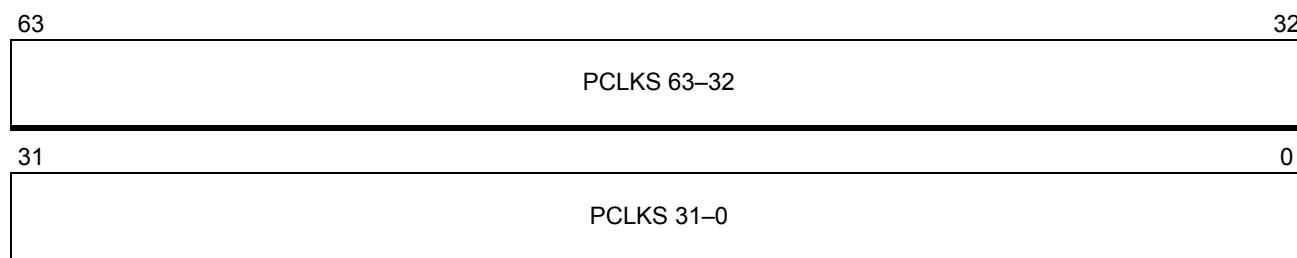
## 13.1.5 Performance Monitoring Registers

### 13.1.5.1 TSC Register

The time-stamp counter (TSC) register maintains a running count of the number of internal processor clock cycles executed after a reset. It is incremented by 1 on each internal processor clock. When the TSC overflows its 64-bit range, it wraps around to 0. See the “Time-Stamp Counter” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

#### TSC Register

**MSR 0010h**



Bit	Mnemonic	Function	R/W	Reset
63–0	PCLKS	Running Clock Cycle Count	R/W	0

#### Field Descriptions

**Processor Clock Cycles (PCLKS)**—Bits 63–0. Running count of number of internal processor clock cycles.

## 13.2 AMD Athlon™ 64 processor and AMD Opteron™ Processor Model-Specific Registers

Table 82 is a listing of the model-specific registers supported by the AMD Athlon™ 64 processor and AMD Opteron™ processor, presented in ascending hexadecimal address order. Register descriptions follow the table, organized according to the following functions:

- Features (see page 368)
- Identification (see page 375)
- Memory typing (see page 375)
- I/O range registers (see page 376)
- System call extension registers (see page 378)
- Segmentation (see page 380)

- System management (see page 219)
- Power management (see page 382)

**Table 82. AMD Athlon™ 64 Processor and AMD Opteron™ Processor MSRs**

Address	Register Name	Description
C000_0080h	EFER	page 368
C000_0081h	STAR	page 378
C000_0082h	LSTAR	page 379
C000_0083h	CSTAR	page 379
C000_0084h	SF_MASK	page 380
C000_0100h	FS.Base	page 380
C000_0101h	GS.Base	page 381
C000_0102h	KernelGSbase	page 381
C001_0000h–C001_0003h	PerfEvtSel <i>i</i>	page 296
C001_0004h–C001_0007h	PerfCtr <i>i</i>	page 295
C001_0010h	SYSCFG	page 369
C001_0015h	HWCR	page 371
C001_0016h, C001_0018h	IORRBase[1:0]	page 376
C001_0017h, C001_0019h	IORRMask[1:0]	page 377
C001_001Ah	TOP_MEM	page 375
C001_001Dh	TOP_MEM2	page 376
C001_001Eh	MANID	page 375
C001_001Fh	NB_CFG	page 373
C001_0041h	FIDVID_CTL	page 382
C001_0042h	FIDVID_STATUS	page 384
C001_0044–C001_0048h	MCi_CTL_MASK	page 196
C001_0050–C001_0053h	IOTRAP_ADDR <i>i</i>	page 386
C001_0054h	IOTRAP_CTL	page 387
C001_0055h	Interrupt Pending Message	page 388
C001_0111h	SMM_BASE	page 225

**Table 82. AMD Athlon™ 64 Processor and AMD Opteron™ Processor MSRs**

C001_0112h	SMM_ADDR	page 230
C001_0113h	SMM_MASK	page 229

## 13.2.1 Feature Registers

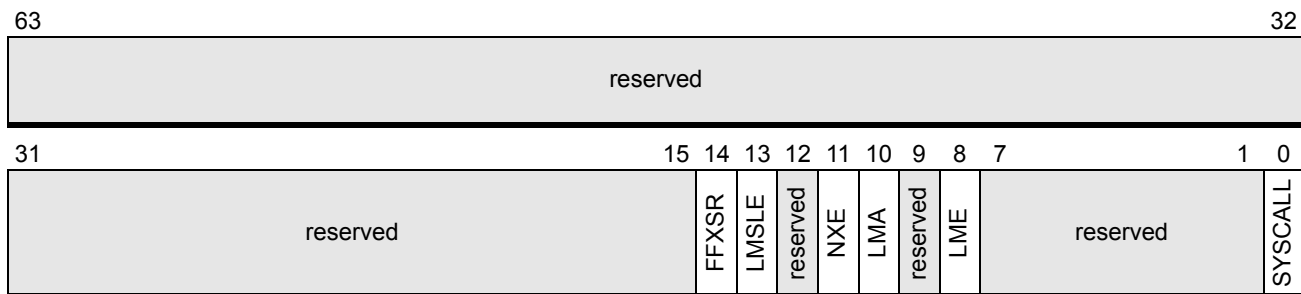
### 13.2.1.1 EFER Register

This register controls which extended features are enabled. See the “Extended Feature Enable Register (EFER)” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

The LMA bit is a read-only status bit. When writing EFER, the processor will signal a #GP(0) if an attempt is made to change LMA from its previous value. BIOS should not modify this register unless it needs to use one of the features enabled by this register.

#### EFER Register

**MSR C000\_0080h**



Bit	Mnemonic	Function	R/W	Reset
63–15	reserved	MBZ	R/W	0
14	FFXSR	Fast FXSAVE/FRSTOR Enable	R/W	0
13	LMSLE	Long Mode Segment Limit Enable	R/W	0
12	reserved	MBZ	R	0
11	NXE	No-Execute Page Enable	R/W	0
10	LMA	Long Mode Active	R	0
9	reserved	MBZ	R	0
8	LME	Long Mode Enable	R/W	0
7–1	reserved	RAZ	R	0
0	SYSCALL	System Call Extension Enable	R/W	0

#### Field Descriptions

**System Call Extension Enable (SCE)**—Bit 0. Enables the system call extension.



**Long Mode Enable (LME)**—Bit 8. Enables the long mode feature.

**Long Mode Active (LMA)**—Bit 10. Indicates the long mode feature is active.

**No-Execute Page Enable (NXE)**—Bit 11. Enables the no-execute page feature.

**Long Mode Segment Limit Enable (LMSLE)**—Bit 13. Enables the long mode segment limit check mechanism. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Fast FXSAVE/FRSTOR Enable (FFXSE)**—Bit 14. Enables the fast FXSAVE/FRSTOR mechanism. A 64-bit operating system uses EDX bit 24 as returned by CPUID instruction standard function 1 to determine the presence of this feature before enabling it. This bit is set once by the operating system and its value is not changed afterwards. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

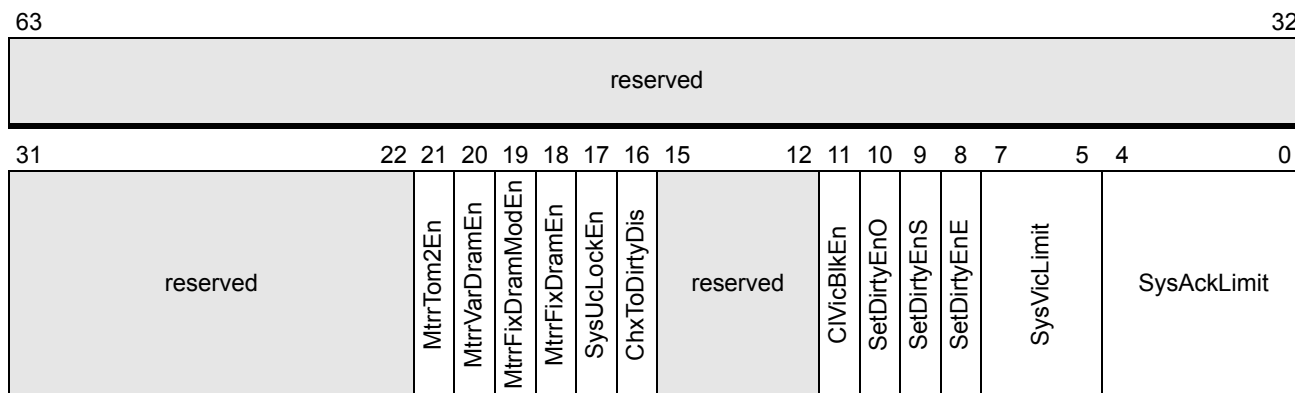
### 13.2.1.2 SYSCFG Register

This register controls the system configuration.

The MtrrFixDramModEn bit should be set to 1 during BIOS initialization of the fixed MTRRs, then cleared to 0 for operation.

#### SYSCFG Register

MSR C001\_0010h



Bit	Mnemonic	Function	R/W	Reset	BIOS
63–22	reserved	RAZ	R	0	
21	MtrrTom2En	Top of Memory Address Register 2 Enable (reserved)	R/W	0	0
20	MtrrVarDramEn	Top of Memory Address Register and I/O Range Register Enable	R/W	0	1
19	MtrrFixDramModEn	RdDram and WrDram Bits Modification Enable	R/W	0	0
18	MtrrFixDramEn	Fixed RdDram and WrDram Attributes Enable	R/W	0	1
17	SysUcLockEn	System Interface Lock Command Enable	R/W	1	1

Bit	Mnemonic	Function	R/W	Reset	BIOS
16	ChxToDirtyDis	Change to Dirty Command Disable	R/W	0	0
15–12	reserved	RAZ	R	0	
11	CIvicBlkEn	Revision B and earlier revisions: CIVicBlk System Interface Command Enable Revision C: Reserved	R/W	0	0
10	SetDirtyEnO	CleanToDirty Command for O->M State Transition Enable	R/W	1	1
9	SetDirtyEnS	SharedToDirty Command for S->M State Transition Enable	R/W	1	1
8	SetDirtyEnE	SharedToDirty Command for E->M State Transition Enable	R/W	0	0
7–5	SysVicLimit	Outstanding Victim Bus Command Limit	R/W	000b	000b
4–0	SysAckLimit	Outstanding Bus Command Limit	R/W	00001b	00001b

## Field Descriptions

**Outstanding Bus Command Limit (SysAckLimit)**—Bit 4–0. Limits maximum number of outstanding bus commands.

**Outstanding Victim Bus Command Limit (SysVicLimit)**—Bit 7–5. Limits maximum number of outstanding victim bus commands.

**SharedToDirty Command for E->M State Transition Enable (SetDirtyEnE)**—Bit 8. Enables generating write probes when transitioning a cache line from Exclusive to Modified.

**SharedToDirty Command for S->M State Transition Enable (SetDirtyEnS)**—Bit 9. Enables generating write probes when transitioning a cache line from Shared to Modified.

**CleanToDirty Command for O->M State Transition Enable (SetDirtyEnO)**—Bit 10. Enables generating write probes when transitioning a cache line from Owned to Modified.

**CIVicBlk System Interface Command Enable (CIvicBlkEn)**—Bit 11. Enables CIVicBlk system interface command.

**Change to Dirty Command Disable (ChxToDirtyDis)**—Bit 16. Disables change to dirty commands, evicts line from DC instead.

**System Interface Lock Command Enable (SysUcLockEn)**—Bit 17. Enables lock commands on system interface.

**Fixed RdDram and WrDram Attributes Enable (MtrrFixDramEn)**—Bit 18. Enables fixed MTRR RdDram and WrDram attributes.

**RdDram and WrDram Bits Modification Enable (MtrrFixDramModEn)**—Bit 19. Enables modification of RdDram and WrDram bits in fixed MTRRs.

**Top of Memory Address Register and I/O Range Register Enable (MtrrVarDramEn)**—Bit 20. Enables use of top of memory address register and the I/O range registers.

**Top of Memory Address Register 2 Enable (MtrrTom2En)**—Bit 21. Enables use of top of memory address register 2 (reserved).

### 13.2.1.3 HWCR Register

This register controls the hardware configuration.

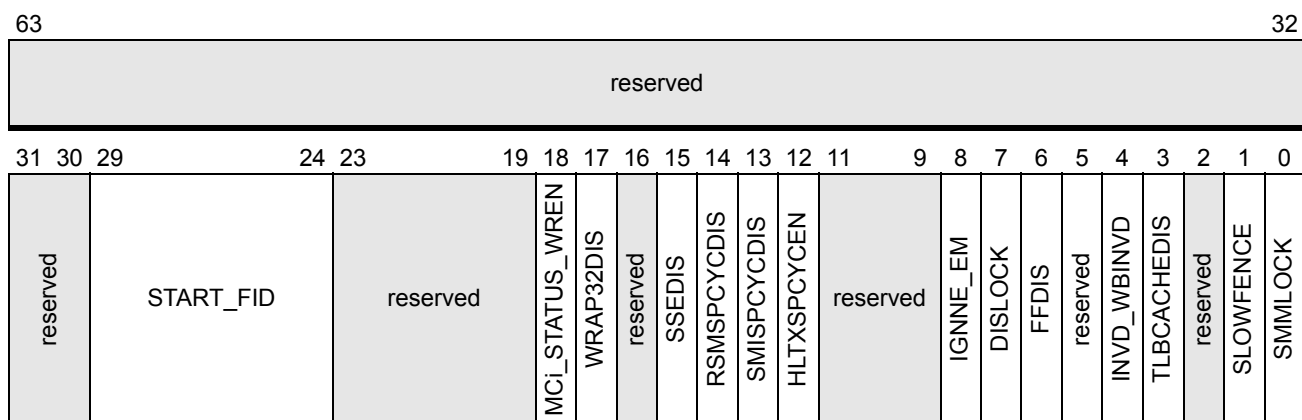
Operating systems that maintain page tables in uncacheable memory (UC memory type) must set the TLBCACHEDIS bit to insure proper operation.

BIOS and SMM handlers may use the WRAP32DIS bit to access physical memory above 4 Gbytes without switching into 64-bit mode. By setting WRAP32DIS to a 1 in conjunction with setting the expanded FS or GS base registers, BIOS can size all of physical memory from legacy mode. To do so, BIOS would write a >32 bit base to the FS or GS base register using WRMSR. Then it would address  $\pm 2$  Gbytes from one of those bases using normal memory reference instructions with a FS or GS override prefix. However, the INVLPG, FST, and SSE store instructions generate 32-bit addresses in legacy mode, regardless of the state of WRAP32DIS.

The MCi\_STATUS\_WREN bit can be used to help debug Machine Check exception handlers. When the MCi\_STATUS\_WREN bit is set, privileged software can write non-zero values to the MCG\_STATUS, MCi\_STATUS, MCi\_ADDR, and if implemented, MCi\_MISC MSRs without generating exceptions, and then simulate a machine check using the INT 18 instruction. Setting a reserved bit in these MSRs does not generate an exception when this mode is enabled. However, setting a reserved bit may result in undefined behavior.

### HWCR Register

**MSR C001\_0015h**



Bit	Mnemonic	Function	R/W	Reset
63–32	reserved	RAZ	R	0
31–30	reserved	SBZ	R/W	0
29–24	START_FID	Startup FID Status	R	0
23–20	reserved	RAZ	R	0
19	reserved	SBZ	R/W	0

Bit	Mnemonic	Function	R/W	Reset
18	MCi_STATUS_WREN	MCi Status Write Enable	R/W	0
17	WRAP32DIS	32-bit Address Wrap Disable	R/W	0
16	reserved			0
15	SSEDIS	SSE Instructions Disable	R/W	0
14	RSMSPCYCDIS	Special Bus Cycle On RSM Disable	R/W	0
13	SMISPCYCDIS	Special Bus Cycle On SMI Disable	R/W	0
12	HLTXSPCYCEN	Enable Special Bus Cycle On Exit From HLT	R/W	0
11–9	reserved	SBZ	R/W	0
8	IGNNE_EM	IGNNE Port Emulation Enable	R/W	0
7	DISLOCK	Disable x86 LOCK prefix functionality	R/W	0
6	FFDIS	TLB Flush Filter Disable	R/W	0
5	reserved	SBZ	R/W	0
4	INVD_WBINVD	INVD to WBINVD Conversion	R/W	0
3	TLBCACHEDIS	Cacheable Memory Disable	R/W	0
2	reserved	SBZ	R/W	0
1	SLOWFENCE	Slow SFENCE Enable	R/W	0
0	SMMLOCK	SMM Code Lock	R/W	0

## Field Descriptions

**SMM Code Lock (SMMLOCK)**—Bit 0. See 6.11.6 “Locking SMM” for the functional description of this bit.

**Slow SFENCE Enable (SLOWFENCE)**—Bit 1. Enable slow sfence.

**Cacheable Memory Disable (TLBCACHEDIS)**—Bit 3. Disable performance improvement that assumes that the PML4, PDP, PDE and PTE entries are in cacheable memory. If page tables are uncachable, TLBCACHEDIS must be set to ensure correct functionality.

**INVD to WBINVD Conversion (INVD\_WBINVD)**—Bit 4. Convert INVD to WBINVD.

**TLB Flush Filter Disable (FFDIS)**—Bit 6. Disable TLB flush filter.

**Disable LOCK (DISLOCK)**—Bit 7. Disable x86 LOCK prefix functionality.

**IGNNE Port Emulation Enable (IGNNE\_EM)**—Bit 8. Enable emulation of IGNNE port.

**Special Bus Cycle On HLT Exit Enable (HLTXSPCYCEN)**—Bit 12. Enables special bus cycle generation on exit from HLT. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Special Bus Cycle On SMI Disable (SMISPCYCDIS)**—Bit 13. Disables special bus cycle on SMI.

**Special Bus Cycle On RSM Disable (RSMSPCYCDIS)**—Bit 14. Disables special bus cycle on RSM.

**SSE Instructions Disable (SSEDIS)**—Bit 15. Disables SSE instructions.

**32-bit Address Wrap Disable (WRAP32DIS)**—Bit 17. Disable 32-bit address wrapping.

**MCi\_STATUS Write Enable (MCi\_STATUS\_WREN)**—Bit 18. When set, writes by software to MCi\_STATUS MSRs do not cause general protection faults. Such writes update all implemented bits in these registers. When clear, writing a non-zero pattern to these registers causes a general protection fault. See the description above for more details. **Startup FID Status (START\_FID)**—Bits 29–24. Status of the startup FID.

### 13.2.1.4 NB\_CFG Register

Software must perform a read-modify-write to this register to change its value.

#### NB\_CFG Register

**MSR C001\_001Fh**



Bit	Mnemonic	Function	R/W	Reset
63–55	reserved		R/W	0
54	InitApicIdCpuIdLo	Initial APIC ID CPU ID Low	R/W	0
53–46	reserved		R/W	0
45	DisUsSysMgtRqToNLdt	Disable Upstream System Management Re-broadcast	R/W	0
44	reserved		R/W	0
43	DisThmIPfMonSmiinterrupts	Disable Performance Monitor SMI	R/W	0
42–37	reserved		R/W	0
36	DisDatMsk	Disable Data Mask	R/W	0
35–32	reserved		R/W	0
31	DisCohLdtCfg	Disable Coherent HyperTransport Configuration Accesses	R/W	0

Bit	Mnemonic	Function	R/W	Reset
30–10	reserved		R/W	0
9	EnRefUseFreeBuf	Enable Display Refresh to Use Free List Buffers	R/W	0
8–0	reserved		R/W	0

## Field Descriptions

**Enable Display Refresh to Use Free List Buffers (EnRefUseFreeBuf)**—Bit 9. For revision CG and earlier revisions, setting this bit to 1 enables display refresh request to use free list buffers. For revision D and later revisions setting this bit to 0 enables display refresh request to use free list buffers. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Disable Coherent HyperTransport Configuration Accesses (DisCohLdtCfg)**—Bit 31. Disables automatic routing of PCI configuration accesses to the processor configuration registers. When set, PCI configuration space accesses which fall within the hard-coded range reserved for AMD Athlon™ 64 and AMD Opteron™ Processor registers (see “Memory System Configuration Registers” on page 39) are instead routed via the configuration address maps. This can be used to effectively hide the configuration registers from software if they are routed to the I/O hub, where they will then get a master abort. It can also be used to provide a means for an external chip to route processor configuration accesses according to some scheme other than the hard-coded version described in “Memory System Configuration Registers” on page 39. When used, this bit needs to be set on all processors in a system. PCI configuration accesses should not be generated if this bit is not set on all processors.

**Disable Data Mask (DisDatMsk)**—Bit 36. Disables DRAM data masking function. For all sized write requests a DRAM read is performed before writing the data. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Disable Performance Monitor SMI (DisThmlPfMonSmiIntr)**—Bit 43. Disables SMI generation for Performance Monitor interrupts. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Disable Upstream System Management Rebroadcast (DisUsSysMgtRqToNLdt)**—Bit 45. Disables re-broadcast of Upstream StpClk and Legacy Input System Management commands downstream. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Initial APIC ID CPU ID Low (InitApicIdCpuIdLo)**—Bit 54. When this bit is set, CpuId and NodeId[2:0] bit field positions are swapped in the APICID. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

0: APICID = {CpuId, NodeId[2:0]}

1: APICID = {NodeId[2:0], CpuId}

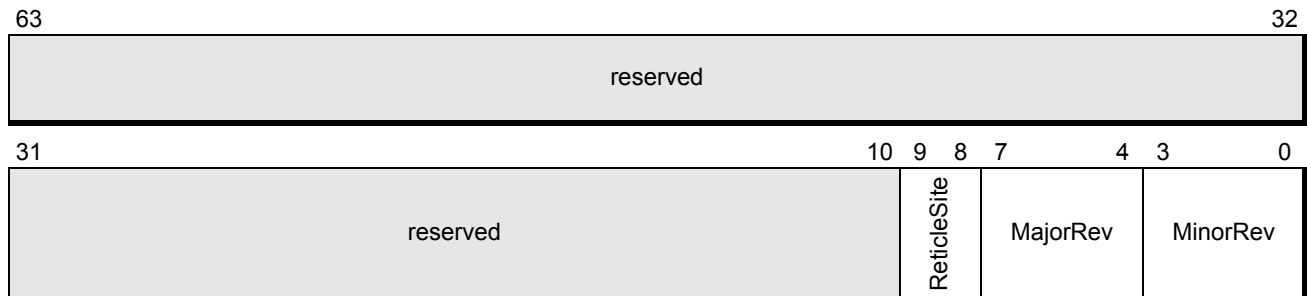
## 13.2.2 Identification Registers

### 13.2.2.1 MANID Register

This status register holds the mask manufacturing identification number.

#### MANID Register

MSR C001\_001Eh



Bit	Mnemonic	Function	R/W
63–10	reserved		
9–8	ReticleSite	Reticle site	R
7–4	MajorRev	Major mask set revision number	R
3–0	MinorRev	Minor mask set revision number	R

#### Field Descriptions

**Minor Mask Set Revision Number (MinorRev)**—Bits 3–0.

**Major Mask Set Revision Number (MajorRev)**—Bits 7–4.

**Reticle Site (ReticleSite)**—Bits 9–8.

## 13.2.3 Memory Typing Registers

### 13.2.3.1 TOP\_MEM Register

This register holds the address of the top of memory. When enabled, this address indicates the first byte of I/O above DRAM.

#### TOP\_MEM Register

MSR C001\_001Ah



31	23	22	0
TOM 8–0		reserved	

Bit	Mnemonic	Function	R/W	Reset
63–40	reserved	RAZ		
39–23	TOM	Top of Memory	R/W	X
22–0	reserved	RAZ		

## Field Descriptions

**Top of Memory (TOM)**—Bits 39–23. Address of top of memory (8-Mbyte granularity).

### 13.2.3.2 TOP\_MEM2 Register

This register holds the second top of memory address. When enabled by setting the MtrrTom2En bit in the SYSCFG register, this address indicates the first byte of I/O above a second allocation of DRAM that starts at 4 Gbytes. Hence, the address in TOP\_MEM2 should be set above 4 Gbytes.

#### TOP\_MEM2 Register

MSR C001\_001Dh

63	40	39	32
reserved			TOM2 16–9
31	23	22	0
TOM2 8–0		reserved	

Bit	Mnemonic	Function	R/W	Reset
63–40	reserved	RAZ		
39–23	TOM2	Second Top of Memory	R/W	
22–0	reserved	RAZ		

## Field Descriptions

**Second Top of Memory (TOM2)**—Bits 39–23. Address of second top of memory (8-Mbyte granularity).

## 13.2.4 I/O Range Registers

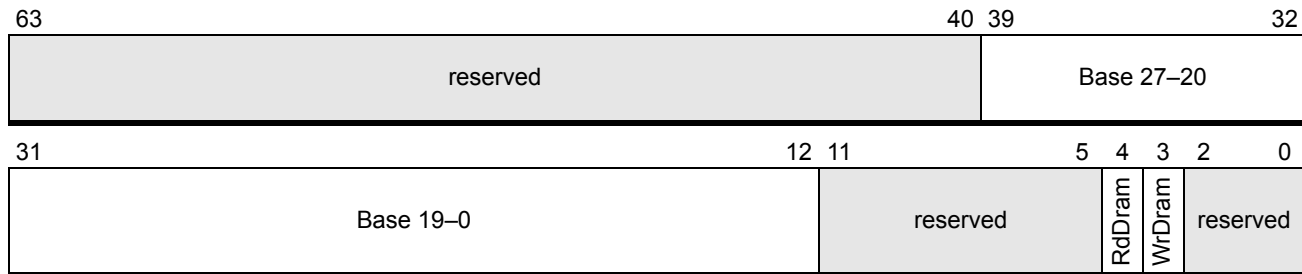
### 13.2.4.1 IORRBase*i* Registers

These registers hold the bases of the variable I/O ranges.



## IORRBase0–1 Registers

MSRs C001\_0016h, C001\_0018h



Bit	Mnemonic	Function	R/W	Reset
63–40	reserved	RAZ		
39–12	Base	Base address	R/W	
11–5	reserved	RAZ		
4	RdDram	Read from DRAM	R/W	
3	WrDram	Write to DRAM	R/W	
2–0	reserved	RAZ		

## Field Descriptions

**Write to DRAM (WrDram)**—Bit 3. If set, stores write to DRAM, otherwise I/O for this range.

**Read from DRAM (RdDram)**—Bit 4. If set, fetches read from DRAM, otherwise from I/O for this range.

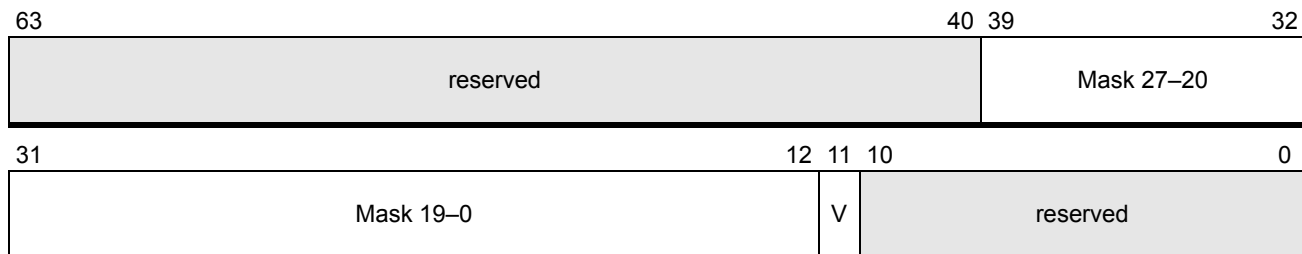
**Base Address (Base)**—Bits 39–12. Base address for this range.

### 13.2.4.2 IORRMask*i* Registers

These registers hold the masks of the variable I/O ranges.

## IORRMask0–1 Registers

MSRs C001\_0017h, C001\_0019h



Bit	Mnemonic	Function	R/W	Reset
63–40	reserved	RAZ		
39–12	Mask	Address mask	R/W	

Bit	Mnemonic	Function	R/W	Reset
11	V	Enables variable I/O range registers	R/W	
10–0	reserved	RAZ		

## Field Descriptions

**Variable I/O Range (V)**—Bit 11. Enables variable I/O range register.

**Address Mask (Mask)**—Bits 39–12. Address mask.

## 13.2.5 System Call Extension Registers

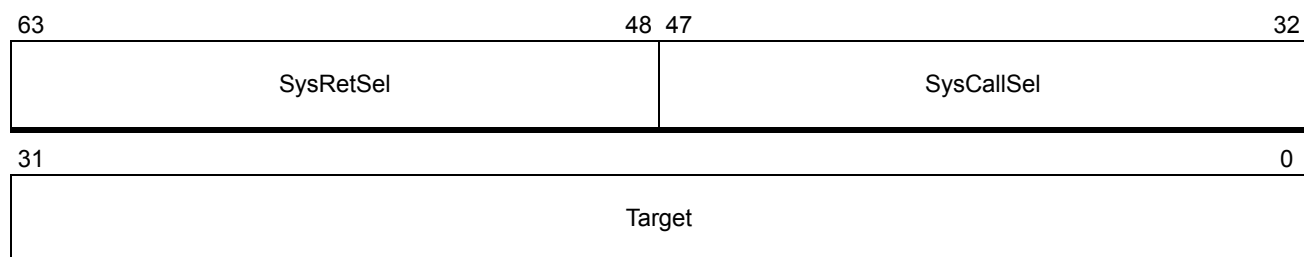
The system call extension is enabled by setting bit 0 in the EFER register. This feature adds the SYSCALL and SYSRET instructions which can be used in flat addressed operating systems as low latency system calls and returns. See the “SYSCALL and SYSRET” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information.

### 13.2.5.1 STAR Register

This register holds the target address used by the SYSCALL instruction and the code and stack segment selector bases used by the SYSCALL and SYSRET instructions.

#### STAR Register

**MSR C000\_0081h**



Bit	Mnemonic	Function	R/W	Reset
63–48	SysRetSel	SYSRET CS and SS	R/W	
47–32	SysCallSel	SYSCALL CS and SS	R/W	
31–0	Target	SYSCALL target address	R/W	

## Field Descriptions

**SYSCALL Target Address (Target)**—Bits 31–0.

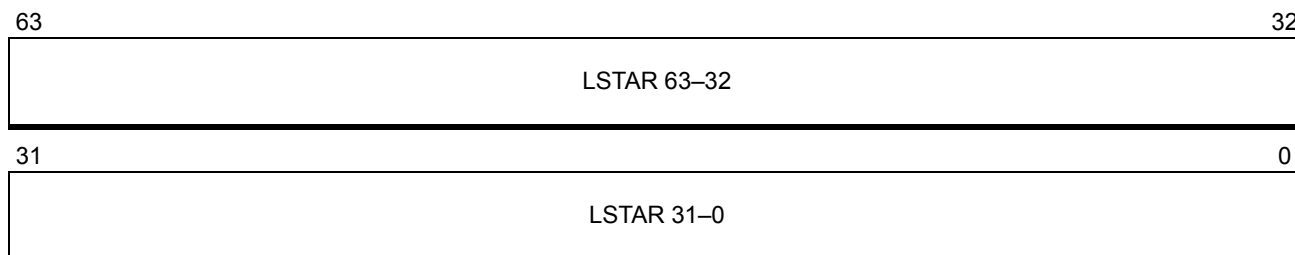
**SYSCALL CS and SS (SysCallSel)**—Bits 47–32.

**SYSRET CS and SS (SysRetSel)**—Bits 63–48.

### 13.2.5.2 LSTAR Register

The address stored in this register must be in canonical form (if not canonical, a #GP fault will occur).

#### LSTAR Register

**MSR C000\_0082h**

Bit	Mnemonic	Function	R/W	Reset
63-0	LSTAR	Long Mode Target Address	R/W	

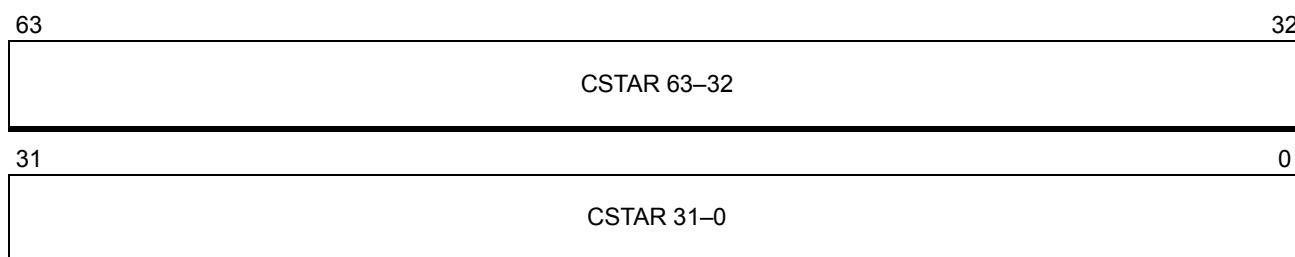
#### Field Descriptions

**Long Mode Target Address (LSTAR)**—Bits 63-0. Target address for 64-bit mode calling programs.

### 13.2.5.3 CSTAR Register

The address stored in this register must be in canonical form (if not canonical, a #GP fault will occur).

#### CSTAR Register

**MSR C000\_0083h**

Bit	Mnemonic	Function	R/W	Reset
63-0	CSTAR	Compatibility mode target address	R/W	

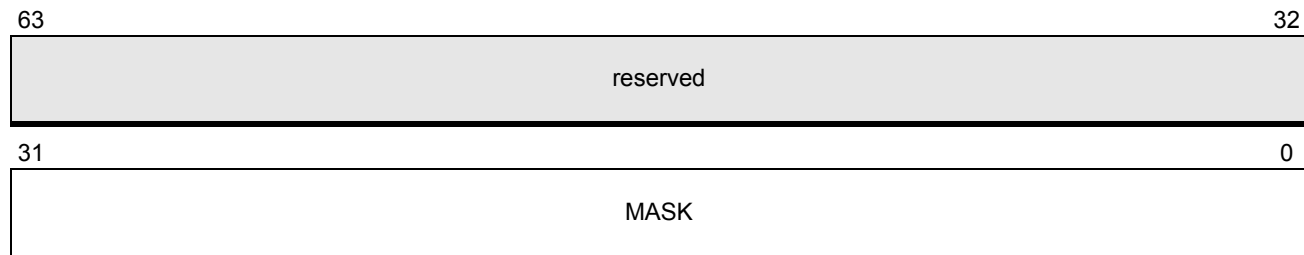
#### Field Descriptions

**Compatibility Mode Target Address (CSTAR)**—Bits 63-0. Target address for compatibility mode calling programs.

### 13.2.5.4 SF\_MASK Register

This register holds the EFLAGS mask used by the SYSCALL instruction. Each one in this mask will clear the corresponding EFLAGS bit when executing the SYSCALL instruction.

#### SF\_MASK Register

**MSR C000\_0084h**

Bit	Mnemonic	Function	R/W	Reset
63–32	reserved	RAZ		
31–0	MASK	SYSCALL Flag Mask	R/W	

#### Field Descriptions

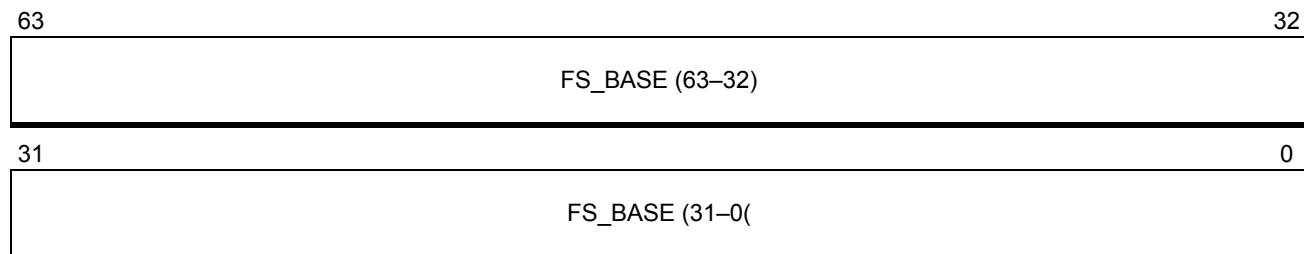
**SYSCALL Flag Mask (MASK)**—Bits 31–0.

### 13.2.6 Segmentation Registers

#### 13.2.6.1 FS.Base Register

This register provides access to the expanded 64 bit FS segment base. The address stored in this register must be in canonical form (if not canonical, a #GP fault will occur).

#### FS.Base Register

**MSR C000\_0100h**

Bit	Mnemonic	Function	R/W	Reset
63–0	FS_BASE	Expanded FS segment base	R/W	

## Field Descriptions

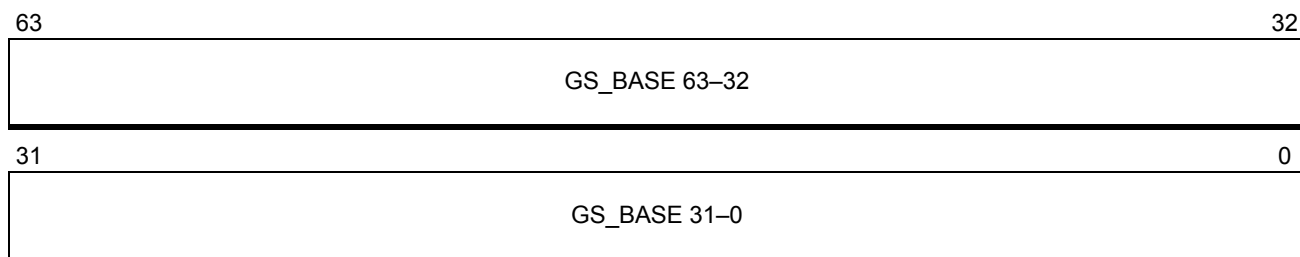
**Expanded FS Segment Base (FS\_BASE)**—Bits 63–0.

### 13.2.6.2 GS.Base Register

This register provides access to the expanded 64 bit GS segment base. The address stored in this register must be in canonical form (if not canonical, a #GP fault will occur).

#### GS.Base Register

**MSR C000\_0101h**



Bit	Mnemonic	Function	R/W	Reset
63–0	GS_BASE	Expanded GS segment base	R/W	

## Field Descriptions

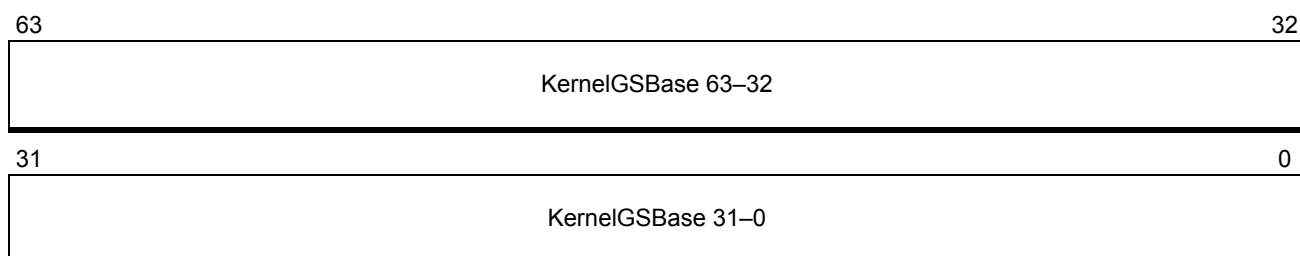
**Expanded GS Segment Base (GS\_BASE)**—Bits 63–0.

### 13.2.6.3 KernelGSbase Register

This register holds the kernel data structure pointer which can be swapped with the GS\_BASE register using the new 64-bit mode instruction SwapGS. See the “SWAPGS Instruction” section in Volume 2 of the *AMD64 Architecture Programmer's Manual* for more information. The address stored in this register must be in canonical form (if not canonical, a #GP fault will occur).

#### KernelGSbase Register

**MSR C000\_0102h**



Bit	Mnemonic	Function	R/W	Reset
63–0	KernelGSBase	Kernel data structure pointer	R/W	

## Field Descriptions

**Kernel Data Structure Pointer (KernelGSBase)—Bits 63–0.**

### 13.2.7 Power Management Registers

Mobile processors contain AMD PowerNow!™ technology hardware that allows the processor operating voltage and frequency to be dynamically controlled for power management purposes. Use CPUID function 8000\_0007h (Get Advanced Power Management Feature Flags) to determine the thermal and power management capabilities of the processor. Refer to “Processor Performance States” on page 265 regarding the use of the FIDVID\_STATUS and FIDVID\_CTL MSRs.

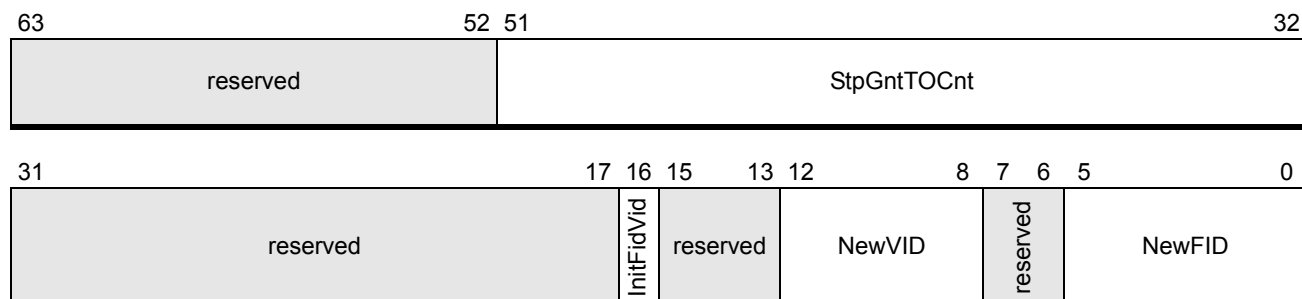
#### 13.2.7.1 FIDVID\_CTL Register

This register holds configuration information relating to power management control. Accessing this register is allowed if the device supports either FID control or VID control as indicated by CPUID. If neither FID control nor VID control is indicated, accessing the FIDVID\_CTL register will cause a GP# fault.

This register is used to specify new Frequency ID (FID) and/or Voltage ID (VID) codes to which to change on the next FID/VID change. This register is also used to control how long to wait following a FID/VID change before the PLL and voltage regulator are stable at the new FID/VID. This register is persistent through a warm reset and is initialized to 0 on a cold reset.

#### FIDVID\_CTL Register

**MSR C001\_0041h**



Bit	Mnemonic	Function	R/W	Reset
63–52	reserved	MBZ		0
51–32	StpGntTOCnt	Stop Grant Time-Out Count	R/W	0
31–17	reserved	MBZ		0
16	InitFidVid	Initiate FID/VID Change	W	0
15–13	reserved	MBZ		0
12–8	NewVID	New VID	R/W	0
7–6	reserved	MBZ	R	0
5–0	NewFID	New FID	R/W	0

## Field Descriptions

**Stop Grant Time-Out Count (StpGntTOCnt)**—Bits 51-32. This field carries a count of system clock cycles (5 ns) that must elapse from the time a new FID is applied until the time that the PLL is stable at the new FID.

**Initiate FID/VID Change (InitFidVid)**—Bits 16. Writing this bit to a 1 initiates a FID/VID change. In a multiprocessor system, only the bootstrap processor sees this bit written as a 1. Writing this bit to a 1 initiates the FID change special bus cycle; in Revision B and earlier revisions, the VID change special bus cycle was also initiated. This is a write-only bit and always reads as 0. The status of the resulting FID/VID change can be determined from the FIDVID\_STATUS register.

**New VID (NewVID)**—Bits 12–8. This field is the new VID to transition to. If an attempt is made to write a NewVID value that corresponds to a voltage greater than the voltage that MaxVID corresponds to in the FIDVID\_STATUS register then the MaxVID value is written instead. For Revision E if an attempt is made to write a NewVID value that corresponds to a voltage less than the voltage that MinVID corresponds to in the FIDVID\_STATUS register then the MinVID value is written instead. See Table 74 on page 283 for VID values and their corresponding voltages. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**New FID (NewFID)**—Bits 5–0. This field is the new FID to transition to. If an attempt is made to write a NewFID value greater than MaxFID in the FIDVID\_STATUS register then the MaxFID value is written instead. See Table 83 for FID code translations.

**Table 83. FID Code Translations**

Reference Clock Multiplier	6-Bit FID Code	Reference Clock Multiplier	6-Bit FID Code
4x	00_0000b	15x	01_0110b
5x	00_0010b	16x	01_1000b
6x	00_0100b	17x	01_1010b
7x	00_0110b	18x	01_1100b
8x	00_1000b	19X	01_1110b
9x	00_1010b	20X	10_0000b
10x	00_1100b	21X	10_0010b
11x	00_1110b	22x	10_0100b
12x	01_0000b	23x	10_0110b
13x	01_0010b	24x	10_1000b
14x	01_0100b	25x	10_1010b
<b>Note:</b> Encodings not listed are reserved. Using reserved encodings will cause unpredictable behavior.			

### 13.2.7.2 FIDVID\_STATUS Register

This register holds status of the current FID, the current VID, pending changes, and maximum values. Accessing this register is allowed if the device supports either FID control or VID control as indicated via CPUID. If neither FID control nor VID control is indicated, accessing the FIDVID\_STATUS register will cause a GP# fault.

See Table 83 on page 383 for a complete list of FID values and their corresponding reference clock multiplier values. See Table 74 on page 283 for VID values and their corresponding voltages.

#### FIDVID\_STATUS Register

**MSR C001\_0042h**

63	61	60	56	55	53	52	48	47	45	44	40	39	37	36	32
reserved	MinVID				reserved	MaxVID				reserved	StartVID				CurrVID

31	30	29	28	24	23	22	21	16	15	14	13	8	7	6	5	0
FidVidPending	reserved	MaxRampVID				reserved	MaxFID				reserved	StartFID				CurrFID

Bit	Mnemonic	Function	R/W	Reset
63–61	reserved	RAZ	R	0
60–56	MinVID	Min VID	R	0
55–53	reserved	RAZ	R	0
52–48	MaxVID	Max VID	R	0
47–45	reserved	RAZ	R	0
44–40	StartVID	Startup VID	R	0
39–37	reserved	RAZ	R	0
36–32	CurrVID	Current VID	R	0
31	FidVidPending	FID/VID Change Pending	R	0
30–29	reserved	RAZ	R	0
28–24	MaxRampVID	Max Ramp VID	R	0
23–22	reserved	RAZ	R	0
21–16	MaxFID	Max FID	R	0
15–14	reserved	RAZ	R	0
13–8	StartFID	Startup FID	R	0
7–6	reserved	RAZ	R	0
5–0	CurrFID	Current FID	R	0



## Field Descriptions

**Min VID (MinVID)**—Bits 60–56. This field indicates the VID code associated with the minimum voltage software can select. See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this field.

**Max VID (MaxVID)**—Bits 52–48. This field indicates the VID code associated with the maximum voltage software can select.

**Startup VID (StartVID)**—Bits 44–40. This field indicates the startup VID.

**Current VID (CurrVID)**—Bits 36–32. This field indicates the current VID.

**FID/VID Change Pending (FidVidPending)**—Bit 31. This bit indicates that a FID/VID change is pending. It is set to 1 by hardware when the InitFidVid bit is set in the FIDVID\_CTL register. It is cleared by hardware when the FID/VID change has completed.

**Max Ramp VID (MaxRampVID)**—Bit 28–24. This field indicates the max Ramp VID.

**Max FID (MaxFID)**—Bit 21–16. This field indicates the max FID.

**Startup FID (StartFID)**—Bit 13–8. This field indicates the startup FID.

**Current FID (CurrFID)**—Bit 5–0. This field indicates the current FID.

## 13.2.8 IO and Configuration Space Trapping to SMI

IO and configuration space trapping to SMI is a mechanism for executing SMI handler if a specific IO access to one of the specified addresses is detected. Access address and access type checking is done before IO instruction execution. If the access address and access type match one of the specified IO address and access types, the SMI handler is executed, the IO instruction is not executed, and a breakpoint set on the IO instruction is not taken. The IO instruction can be executed inside the SMI handler. This mechanism has higher priority than the SMI interrupt. If the trap is not taken, check for SMI interrupt is done after the IO instruction has been executed.

Configuration accesses are special IO accesses. An IO access is defined as a configuration access when IO instruction address bits 31–0 are CFCh, CFDh, CFEh, or CFFh. The access address for a configuration space access is the current value of doubleword IO register CF8h. The access address for an IO access that is not a configuration access is equivalent to the IO instruction address, bits 31–0.

The access address is compared with SmiAddr, and the instruction access type is compared with the enabled access types defined by ConfigSMI, SmiOnRdEn, and SmiOnWrEn. Access address bits 23–0 can be masked with SmiMask. Fields SmiAddr, SmiMask, ConfigSMI, SmiOnRdEn, and SmiOnWrEn are defined in IOTRAP\_ADDR*i* registers (*i* = 0,1,2,3).

An SMI handler executed as a result of IO and configuration space trapping will by default only be detected by the processor executing the IO instruction. The user has the ability to enable an SMI special bus cycle and RSM special bus cycle generation.

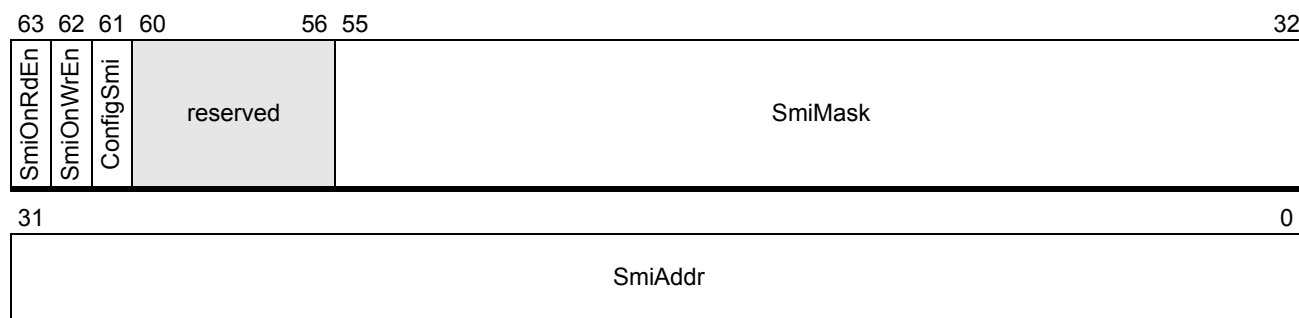
IO and configuration space trapping to SMI applies only to single IO instructions; it does not apply to string and REP IO instructions.

### 13.2.8.1 IOTRAP\_ADDR*i* Registers

See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this register.

#### IOTRAP\_ADDR*i* Registers

MSRs C001\_0050h, C001\_0051h,  
C001\_0052h, C001\_0053h



Bit	Mnemonic	Function	R/W	Reset
63	SmiOnRdEn	Enable SMI on IO Read	R/W	0
62	SmiOnWrEn	Enable SMI on IO Write	R/W	0
61	ConfigSmi	Configuration Space SMI	R/W	0
60-56	reserved	RAZ	R	
55-32	SmiMask	SMI Mask	R/W	0
31-0	SmiAddr	SMI Address	R/W	0

#### Field Descriptions

**Enable SMI on IO Read (SmiOnRdEn)**—Bits 63. Enables SMI generation on a read access.

**Enable SMI on IO Write (SmiOnWrEn)**—Bit 62. Enables SMI generation on a write access.

**Configuration Space SMI (ConfigSmi)**—Bit 61. 1 = configuration access (see “IO and Configuration Space Trapping to SMI” on page 385 for more information on configuration access detection); 0 = IO access different than configuration access.

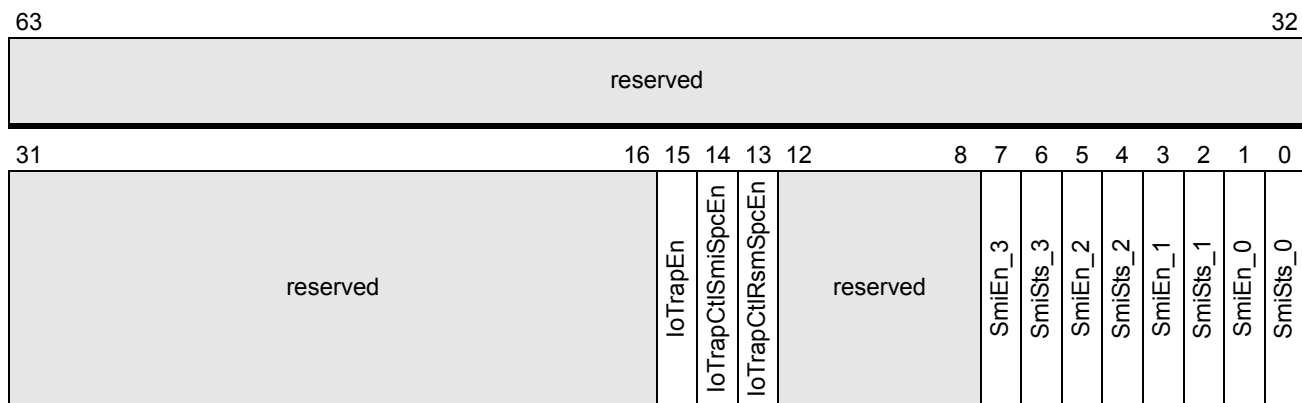
**SMI Mask (SmiMask)**—Bits 55-32. SMI IO trap mask. 0 = mask address bit; 1 = do not mask address bit. When the ConfigSmi bit is set bits 33 and 32 are ignored.

**SMI Address (SmiAddr)**—Bits 31-0. SMI IO trap address. When the ConfigSmi bit set is the SMI address is internally doubleword aligned and bits 0 and 1 of SmiAddr are ignored.

### 13.2.8.2 IOTRAP\_CTL Register

See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this register.

#### IOTRAP\_CTL Register

**MSR C001\_0054h**

Bit	Mnemonic	Function	R/W	Reset
63–16	reserved	RAZ	R	
15	IoTrapEn	IO Trap Enable	R/W	0
14	IoTrapCtlSmiSpcEn	IO Trap Control SMI Special Cycle Enable	R/W	0
13	IoTrapCtlRsmSpcEn	IO Trap Control RSM Special Cycle Enable	R/W	0
12–8	reserved	RAZ	R	
7	SmiEn_3	SMI Enable 3	R/W	0
6	SmiSts_3	SMI Status 3	R/W	0
5	SmiEn_2	SMI Enable 2	R/W	0
4	SmiSts_2	SMI Status 2	R/W	0
3	SmiEn_1	SMI Enable 1	R/W	0
2	SmiSts_1	SMI Status 1	R/W	0
1	SmiEn_0	SMI Enable 0	R/W	0
0	SmiSts_0	SMI Status 0	R/W	0

#### Field Descriptions

**IO Trap Enable (IoTrapEn)**—Bit 15. Enable IO and configuration space trapping.

**IO Trap Control SMI Special Cycle Enable (IoTrapCtlSmiSpcEn)**—Bit 14. Enable SMI special bus cycle generation when SMI handler is entered as a result of IO and configuration space trapping.

**IO Trap Control RSM Special Cycle Enable (IoTrapCtlRsmSpcEn)**—Bit 13. Enable RSM special bus cycle generation when SMI handler is entered as a result of IO and configuration space trapping.

**SMI Enable 3 (SmiEn\_3)**—Bit 7. Enable SMI generation if IO access matches access specified in MSR C001\_0053h.

**SMI Status 3 (SmiSts\_3)**—Bit 6. Set if IO access matched access specified in MSR C001\_0053h.

**SMI Enable 2 (SmiEn\_2)**—Bit 5. Enable SMI generation if IO access matches access specified in MSR C001\_0052h.

**SMI Status 2 (SmiSts\_2)**—Bit 4. Set if IO access matched access specified in MSR C001\_0052h.

**SMI Enable 1 (SmiEn\_1)**—Bit 3. Enable SMI generation if IO access matches access specified in MSR C001\_0051h.

**SMI Status 1 (SmiSts\_1)**—Bit 2. Set if IO access matched access specified in MSR C001\_0051h.

**SMI Enable 0 (SmiEn\_0)**—Bit 1. Enable SMI generation if IO access matches access specified in MSR C001\_0050h.

**SMI Status 0 (SmiSts\_0)**—Bit 0. Set if IO access matched access specified in MSR C001\_0050h.

### 13.2.9 Interrupt Pending Message Register

This register is used to ensure that the system I/O Hub can wake the processor out of the stop grant state when there is a pending interrupt. If the interrupt pending message is not specified by this register, it is possible for the system I/O Hub to place the processor into the stop grant state while an interrupt is pending in the processor. This register must be setup for platforms that support the C2 or C3 power management states.

This register enables the processor to send a message to the I/O hub that results in the pending interrupt being serviced. The two message types are the IO space message and the HyperTransport INT\_PENDING message. HyperTransport INT\_PENDING message is defined by the HyperTransport 1.05c specification.

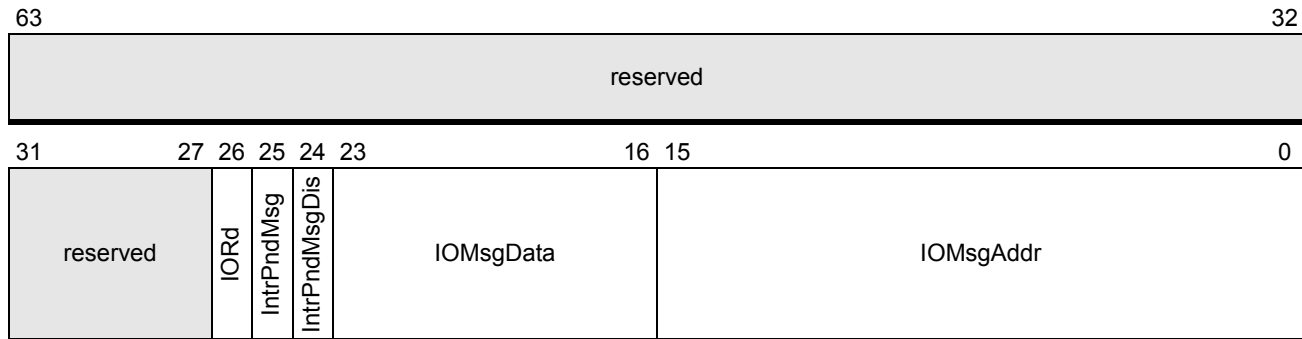
If the processor or the I/O hub does not support the INT\_PENDING HyperTransport message, the IO space message should be selected by `IntPndMsg`. A check for a pending interrupt is performed at the end of an IO instruction. If there is a pending interrupt and `STPCLK` is asserted, the processor executes a byte-size read or write to IO space defined with `IORD`, `IOMsgAddr`, and `IOMsgData` (used only for IO writes) to generate an SMI. The SMI wakes up the processor so the original pending interrupt can be serviced. The SMI handler should not take any action if the SMI is generated by this mechanism. In order to prevent SMI generation with this mechanism in the SMI handler, `IntrPndMsgDis` bit should be set in the SMI handler before the first IO instruction is executed, and it should be cleared prior to resuming from SMM.

If the processor and the I/O hub support the INT\_PENDING HyperTransport message, it should be selected by `IntPndMsg`. The check for a pending interrupt is performed when entering the stop grant state. If there is a pending interrupt, the processor sends an INT\_PENDING HyperTransport message to the I/O hub.

See “Register Differences in Revisions of the AMD Athlon™ 64 and AMD Opteron™ Processors” on page 29 for revision information about this register.

## Interrupt Pending Message Register

MSR C001\_0055h



Bit	Mnemonic	Function	R/W	Reset
63–27	reserved	RAZ	R	
26	IORd	IO Read	R/W	0
25	IntrPndMsg	Interrupt Pending Message	R/W	0
24	IntrPndMsgDis	Interrupt Pending Message Disable	R/W	0
23–16	IOMsgData	IO Message Data	R/W	0
15–0	IOMsgAddr	IO Message Address	R/W	0

## Field Descriptions

**IO Read (IORd)**—Bit 26. 1 = IO read; 0 = IO write.

**Interrupt Pending Message (IntrPndMsg)**—Bit 25. Select an interrupt pending message type. 0 = HyperTransport INT\_PENDING message; 1 = IO space message. This bit should be set after IORd, IOMsgAddr and IOMsgData fields are programmed.

**Interrupt Pending Message Disable (IntrPndMsgDis)**—Bit 24. Disable generating an interrupt pending message defined with IntrPndMsg.

**IO Message Data (IOMsgData)**—Bits 23–16. IO space message data. This field is only used when an IO write is selected (IORd=0).

**IO Message Address (IOMsgAddr)**—Bits 15–0. IO space message address.



# Glossary

---

**128-bit media instructions.** Instructions that use the 128-bit XMM registers. These are a combination of SSE and SSE2 instruction sets.

**64-bit media instructions.** Instructions that use the 64-bit MMX™ registers. These are primarily a combination of MMX and 3DNow!™ instruction sets, with some additional instructions from the SSE and SSE2 instruction sets.

**16-bit mode.** Legacy mode or compatibility mode in which a 16-bit address size is active. See *legacy mode* and *compatibility mode*.

**32-bit mode.** Legacy mode or compatibility mode in which a 32-bit address size is active. See *legacy mode* and *compatibility mode*.

**64-bit mode.** A submode of *long mode*. In 64-bit mode, the default address size is 64 bits and new features, such as register extensions, are supported for system and application software.

**absolute.** Said of a displacement that references the base of a code segment rather than an instruction pointer. Contrast with *relative*.

**AH–DH.** The high 8-bit AH, BH, CH, and DH registers. Compare *AL–DL*.

**AL–DL.** The low 8-bit AL, BL, CL, and DL registers. Compare *AH–DH*.

**AL–r15B.** The low 8-bit AL, BL, CL, DL, SIL, DIL, BPL, SPL, and R8B–R15B registers, available in 64-bit mode.

**biased exponent.** The sum of a floating-point value's exponent and a constant bias for a particular floating-point data type. The bias makes the range of the biased exponent always positive, which allows reciprocation without overflow.

**byte.** Eight bits.

**clear.** To write a bit-value of 0. Compare *set*.

**compatibility mode.** A submode of *long mode*. In compatibility mode, the default address size is 32 bits and legacy 16-bit and 32-bit applications run without modification.

**b.** A bit, as in *1 Mb* for one megabit, or *lsb* for least-significant bit.

**B.** A byte, as in *1 MB* for one megabyte, or *LSB* for least-significant byte.

**canonical address.** AMD Athlon™ 64 and AMD Opteron™ Processors implement 48-bit virtual addresses. A virtual address having all 1s or all 0s in bit 63 through the most significant implemented bit (bit 47 for AMD Athlon™ 64 processor and AMD Opteron™ processor) is said to be in canonical form. Accessing a virtual address that is not in canonical form results in a fault.

**commit.** To irreversibly write, in program order, an instruction's result to software-visible storage, such as a register (including flags), the data cache, an internal write buffer, or memory.

**CPL.** Current privilege level.

**CR $n$ .** Control register number  $n$ .

**CS.** Code segment register.

**direct.** Referencing a memory location whose address is included in the instruction's syntax as an immediate operand. The address may be an absolute or relative address. Compare *indirect*.

**dirty data.** Data held in the processor's caches or internal buffers that is more recent than the copy held in main memory.

**displacement.** A signed value that is added to the base of a segment (absolute addressing) or an instruction pointer (relative addressing). Same as *offset*.

**doubleword.** Two words, or four bytes, or 32 bits.

**double quadword.** Eight words, or 16 bytes, or 128 bits. Also called *octword*.

**DP.** Two processors or dual processors.

**eAX–eSP.** The 16-bit AX, BX, CX, DX, DI, SI, BP, SP registers or the 32-bit EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP registers. Compare *rAX–rSP*.

**EFER.** Extended features enable register.

**effective address size.** The address size for the current instruction after accounting for the default address size and any address-size override prefix.

**effective operand size.** The operand size for the current instruction after accounting for the default operand size and any operand-size override prefix.

**eFLAGS.** 16-bit or 32-bit flags register. Compare *rFLAGS*.

**EFLAGS.** 32-bit (extended) flags register.

**eIP.** 16-bit or 32-bit instruction-pointer register. Compare *rIP*.

**EIP.** 32-bit (extended) instruction-pointer register.

**element.** See *vector*.

**exception.** An abnormal condition that occurs as the result of executing an instruction. The processor's response to an exception depends on the type of the exception. For all exceptions except 128-bit media SIMD floating-point exceptions and x87 floating-point exceptions, control is transferred to the handler (or service routine) for that exception, as defined by the exception's vector. For floating-point exceptions defined by the IEEE 754 standard, there are both masked and unmasked responses. When unmasked, the exception handler is called, and when masked a default response is provided instead of calling the handler.

**FLAGS.** 16-bit flags register.

**flush.** When referring to caches, (1) to writeback, if modified, and invalidate, as in “flush the cache line”; when referring to the processor pipeline, (2) to invalidate, as in “flush the pipeline”; or (3) to change a value, as in “flush to zero”.

**GDT.** Global descriptor table.



**GDTR.** Global descriptor table register.

**GPRs.** General-purpose registers. For the 16-bit data size, these are AX, BX, CX, DX, DI, SI, BP, SP. For the 32-bit data size, these are EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP. For the 64-bit data size, these include RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, and R8–R15.

**IDT.** Interrupt descriptor table.

**IDTR.** Interrupt descriptor table register.

**IGN.** Ignore. Field is ignored.

**indirect.** Referencing a memory location whose address is in a register or other memory location. The address may be an absolute or relative address. Compare *direct*.

**IP.** 16-bit instruction-pointer register.

**IRB.** The virtual-8086 mode interrupt-redirection bitmap.

**IST.** The long-mode interrupt-stack table.

**IVT.** The real-address mode interrupt-vector table.

**LDT.** Local descriptor table.

**LDTR.** Local descriptor table register.

**legacy x86.** The legacy x86 architecture. See “Related Documents” on page 28 for descriptions of the legacy x86 architecture.

**legacy mode.** An operating mode of the AMD64 architecture in which existing 16-bit and 32-bit applications and operating systems run without modification. A processor implementation of the AMD64 architecture can run in either *long mode* or *legacy mode*. Legacy mode has three submodes, *real mode*, *protected mode*, and *virtual-8086 mode*.

**long mode.** An operating mode unique to the AMD64 architecture. A processor implementation of the AMD64 architecture can run in either *long mode* or *legacy mode*. Long mode has two submodes, *64-bit mode* and *compatibility mode*.

**lsb.** least-significant bit.

**LSB.** least-significant byte.

**main memory.** Physical memory, such as RAM and ROM (but not cache memory) that is installed in a particular computer system.

**mask.** (1) A control bit that prevents the occurrence of a floating-point exception from invoking an exception handling routine. (2) A field of bits used for a control purpose.

**MBZ.** Must be 0. If software attempts to set an MBZ bit to 1, a general-protection exception (#GP) occurs.

**memory.** Unless otherwise specified, *main memory*.

**ModRM.** A byte following an instruction opcode that specifies address calculation based on mode (mod), register (r), and memory (m) variables.

**offset.** A direct memory offset. I.e., a displacement that is added to the base of a code segment (for absolute addressing) or to an instruction pointer (for addressing relative to the instruction pointer, as in RIP-relative addressing).

**MP.** More than two processors.

**msb.** Most-significant bit.

**MSB.** Most-significant byte.

**MSR.** Model-specific register.

**multimedia instructions.** A combination of *128-bit media instructions* and *64-bit media instructions*.

**octword.** Same as *double quadword*.

**offset.** Same as *displacement*.

**overflow.** The condition in which a floating-point number is larger in magnitude than the largest, finite, positive or negative number that can be represented in the data-type format being used.

**packed.** See *vector*.

**PAE.** Physical-address extensions.

**physical memory.** Actual memory, consisting of *main memory* and cache.

**probe.** A check for an address in a processor's caches or internal buffers. *External probes* originate outside the processor, and *internal probes* originate within the processor.

**processor.** This term refers to AMD Athlon™ 64 processor architecture, AMD Opteron™ processor architecture, AMD Sempron™ processor architecture and AMD Turion™ 64 Mobile Technology processor architecture. This document covers all classes of these devices. This term is used to refer to packages that contain one or more processor cores. For details about differences between them, see the relevant processor data sheets and functional data sheets.

**protected mode.** A submode of *legacy mode*.

**quadword.** Four words, or eight bytes, or 64 bits.

**r8–r15.** The 8-bit R8B–R15B registers, or the 16-bit R8W–R15W registers, or the 32-bit R8D–R15D registers, or the 64-bit R8–R15 registers.

**rAX–rSP.** The 16-bit AX, BX, CX, DX, DI, SI, BP, SP registers, or the 32-bit EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP registers, or the 64-bit RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP registers. The “r” variable should be replaced by nothing for 16-bit size, “E” for 32-bit size, or “R” for 64-bit size.

**RAX.** 64-bit version of EAX register.

**RAZ.** Read as zero (0), regardless of what is written.

**RBP.** 64-bit version of EBP register.

**RBX.** 64-bit version of EBX register.

**RCX.** 64-bit version of ECX register.

**RDI.** 64-bit version of EDI register.

**RDX.** 64-bit version of EDX register.

**real-address mode.** See *real mode*.

**real mode.** A short name for *real-address mode*, a submode of *legacy mode*.

**relative.** Referencing with a displacement (also called offset) from an instruction pointer rather than the base of a code segment. Contrast with *absolute*.

**reserved.** Fields marked as reserved may be used at some future time. In order to preserve compatibility with future processors, reserved fields require special handling when read or written by software.

Reserved fields may be further qualified as MBZ, RAZ, SBZ or IGN (see definitions).

Software must not depend on the state of a reserved field, nor upon the ability of such fields to return state previously written.

If a reserved field is not marked with one of the above qualifiers, software shall not change the state of that field; it must reload that field with the same values returned from a prior read.

**REX.** An instruction prefix that specifies a 64-bit operand size and provides access to additional registers.

**rFLAGS.** 16-bit, 32-bit, or 64-bit flags register. Compare *RFLAGS*.

**RFLAGS.** 64-bit flags register. Compare *rFLAGS*.

**rIP.** 16-bit, 32-bit, or 64-bit instruction-pointer register. Compare *RIP*.

**RIP-Relative Addressing.** 4-bit instruction-pointer register. Addressing relative to the 64-bit RIP instruction pointer. Compare *offset*.

**RSI.** 64-bit version of ESI register.

**RSP.** 64-bit version of ESP register.

**SBZ.** Should be zero. If software attempts to set an SBZ bit to 1, it results in undefined behavior.

**set.** To write a bit-value of 1. Compare *clear*.

**SIB.** A byte following an instruction opcode that specifies address calculation based on scale (S), index (I), and base (B).

**SIMD.** Single instruction, multiple data. See *vector*.

**SP.** Stack pointer register.

**SS.** Stack segment register.

**SSE.** Streaming SIMD extensions instruction set. See *128-bit media instructions* and *64-bit media instructions*.

**SSE2.** Extensions to the SSE instruction set. See *128-bit media instructions* and *64-bit media instructions*.

**sticky bit.** A bit that is set or cleared by hardware and that remains in that state until explicitly changed by software.

**TOP.** x87 top-of-stack pointer.

**TPR.** Task-priority register (CR8), a new register introduced in the AMD64 architecture to speed interrupt management.

**TR.** Task register.

**TSS.** Task state segment.

**underflow.** The condition in which a floating-point number is smaller in magnitude than the smallest non-zero, positive or negative number that can be represented in the data-type format being used.

**UP.** One processor or uniprocessor.

**vector.** (1) A set of integer or floating-point values, called *elements*, that are packed into a single operand. Most of the 128-bit and 64-bit media instructions use vectors as operands. Vectors are also called *packed* or *SIMD* (single-instruction multiple-data) operands. (2) An index into an interrupt descriptor table (IDT), used to for accessing exception handlers. Compare *exception*.

**virtual-8086 mode.** A submode of *legacy mode*.

**word.** Two bytes, or 16 bits.

**x86.** See *legacy x86*.

# Index of Register Names

---

## B

### BU Machine Check

- Address Register (MSR 040Ah) 211
- Control Mask Register (MSR C001\_0046h) 210
- Control Register (MSR 0408h) 207
- Status Register (MSR 0409h) 210

## C

- Capabilities Pointer Register (Function 0: Offset 34h) 43
- Class Code/Revision ID Register (Function 0: Offset 08h) 42
- Class Code/Revision ID Register (Function 1: Offset 08h) 69
- Class Code/Revision ID Register (Function 2: Offset 08h) 83
- Class Code/Revision ID Register (Function 3: Offset 08h) 120
- Clear Interrupts Register (Function 3: Offset B4h) 157
- Clock Power/Timing High Register (Function 3: Offset D8h) 160
- Clock Power/Timing Low Register (Function 3: Offset D4h) 157
- Config
  - Base and Limit 0 Register (Function 1: Offset E0h) 78
  - Base and Limit 1 Register (Function 1: Offset E4h) 78
  - Base and Limit 2 Register (Function 1: Offset E8h) 78
  - Base and Limit 3 Register (Function 1: Offset ECh) 78
- Configuration Address Register 0CF8h 37

## D

### DC Machine Check

- Address Register (MSR 0402h) 203
- Control Mask Register (MSR C001\_0044h) 200
- Control Register (MSR 0400h) 199
- Status (MSR 0401h) 200
- Device/Vendor ID register (Function 0: Offset 00h) 41
- Device/Vendor ID Register (Function 1: Offset 00h) 68
- Device/Vendor ID Register (Function 2: Offset 00h) 82
- Device/Vendor ID Register (Function 3: Offset 00h) 119
- Display Refresh Flow Control Buffers 151
- DRAM
  - Bank Address Mapping Register (Function 2: Offset 80h) 88
  - Base 0 Register (Function 1: Offset 40h) 71
  - Base 1 Register (Function 1: Offset 48h) 71
  - Base 2 Register (Function 1: Offset 50h) 71
  - Base 3 Register (Function 1: Offset 58h) 71
  - Base 4 Register (Function 1: Offset 60h) 71
  - Base 5 Register (Function 1: Offset 68h) 71
  - Base 6 Register (Function 1: Offset 70h) 71
  - Base 7 Register (Function 1: Offset 78h) 71
  - Config High Register (Function 2: Offset 94h) 112
  - Config Low Register (Function 2: Offset 90h) 106
  - CS Base 0 Register (Function 2: Offset 40h) 84
  - CS Base 1 Register (Function 2: Offset 44h) 84

CS Base 2 Register (Function 2: Offset 48h) 84  
 CS Base 3 Register (Function 2: Offset 4Ch) 84  
 CS Base 4 Register (Function 2: Offset 50h) 84  
 CS Base 5 Register (Function 2: Offset 54h) 84  
 CS Base 6 Register (Function 2: Offset 58h) 84  
 CS Base 7 Register (Function 2: Offset 5Ch) 84  
 CS Mask 0 Register (Function 2: Offset 60h) 87  
 CS Mask 1 Register (Function 2: Offset 64h) 87  
 CS Mask 2 Register (Function 2: Offset 68h) 87  
 CS Mask 3 Register (Function 2: Offset 6Ch) 87  
 CS Mask 4 Register (Function 2: Offset 70h) 87  
 CS Mask 5 Register (Function 2: Offset 74h) 87  
 CS Mask 6 Register (Function 2: Offset 78h) 87  
 CS Mask 7 Register (Function 2: Offset 7Ch) 87  
 Delay Line Register (Function 2: Offset 98h) 115  
 Limit 0 Register (Function 1: Offset 44h) 72  
 Limit 1 Register (Function 1: Offset 4Ch) 72  
 Limit 2 Register (Function 1: Offset 54h) 72  
 Limit 3 Register (Function 1: Offset 5Ch) 72  
 Limit 4 Register (Function 1: Offset 64h) 72  
 Limit 5 Register (Function 1: Offset 6Ch) 72  
 Limit 6 Register (Function 1: Offset 74h) 72  
 Limit 7 Register (Function 1: Offset 7Ch) 72  
 Scrub Address High Register (Function 3: Offset 60h) 142  
 Scrub Address Low Register (Function 3: Offset 5Ch) 142  
 Timing High Register (Function 2: Offset 8Ch) 104  
 Timing Low Register (Function 2: Offset 88h) 102

## G

### GART

Aperture Base Register (Function 3: Offset 94h) 155  
 Aperture Control Register (Function 3: Offset 90h) 154  
 Cache Control Register (Function 3: Offset 9Ch) 157  
 Table Base Register (Function 3: Offset 98h) 156  
 Global Machine Check Capabilities Register (MSR 0179h) 193  
 Global Machine Check Exception Reporting Control Register (MSR 017Bh) 194  
 Global Machine Check Processor Status Register (MSR 017Ah) 193

## H

### Header

Type Register (Function 0: Offset 0Ch) 43  
 Type Register (Function 1: Offset 0Ch) 69  
 Type Register (Function 2: Offset 0Ch) 83  
 Type Register (Function 3: Offset 0Ch) 120

### HyperTransport

Initialization Control Register (Function 0: Offset 6Ch) 53  
 Read Pointer Optimization Register (Function 3: Offset DCh) 161  
 Transaction Control Register (Function 0: Offset 68h) 48

## I

### IC Machine Check

Address Register (MSR 0406h) 207  
 Control Mask Register (MSR C001\_0045h) 205

Control Register (MSR 0404h) 204

Status Register (MSR 0405h) 206

**L****LDT0**

Buffer Count Register (Function 0: Offset 90h) 62  
 Bus Number Register (Function 0: Offset 94h) 64  
 Capabilities Register (Function 0: Offset 80h) 54  
 Feature Capability Register (Function 0: Offset 8Ch) 61  
 Frequency/Revision Register (Function 0: Offset 88h) 60  
 Link Control Register (Function 0: Offset 84h) 56  
 Type Register (Function 0: Offset 98h) 65

**LDT1**

Buffer Count Register (Function 0: Offset B0h) 62  
 Bus Number Register (Function 0: Offset B4h) 64  
 Capabilities Register (Function 0: Offset A0h) 54  
 Feature Capability Register (Function 0: Offset ACh) 61  
 Frequency/Revision Register (Function 0: Offset A8h) 60  
 Link Control Register (Function 0: Offset A4h) 56  
 Type Register (Function 0: Offset B8h) 65

**LDT2**

Buffer Count Register (Function 0: Offset D0h) 62  
 Bus Number Register (Function 0: Offset D4h) 64  
 Capabilities Register (Function 0: Offset C0h) 54  
 Feature Capability Register (Function 0: Offset CCh) 61  
 Frequency/Revision Register (Function 0: Offset C8h) 60  
 Link Control Register (Function 0: Offset C4h) 56  
 Type Register (Function 0: Offset D8h) 65

**LS Machine Check**

Address Register (MSR 040Eh) 213  
 Control Mask Register (MSR C001\_0047h) 212  
 Control Register (MSR 040Ch) 212  
 Status Register (MSR 040Dh) 213

**M**

MC0\_CTL Register (MSR 0400h) 199

MC0\_STATUS Register (MSR 0401h) 200

MC1\_CTL Register (MSR 0404h) 204

MC2\_CTL Register (MSR 0408h) 207

MC3\_CTL Register (MSR 040Ch) 212

**MCA**

NB Address High Register (Function 3: Offset 54h) 133  
 NB Address Low Register (Function 3: Offset 50h) 133  
 NB Configuration Register (Function 3: Offset 44h) 124  
 NB Control Register (Function 3: Offset 40h) 121  
 NB Status High Register (Function 3: Offset 4Ch) 130  
 NB Status Low Register (Function 3: Offset 48h) 127

MCG\_CAP Register (MSR 0179h) 193

MCG\_CTL Register (MSR 017Bh) 194

MCG\_STATUS Register (MSR 017Ah) 193

MCi\_ADDR Register (MSRs 0402h, 0406h, 040Ah, 040Eh, 0412h) 198

MCi\_CTL Registers (MSRs 0400h, 0404h, 0408h, 040Ch, 0410h) 195

MCi\_CTL\_MASK Registers (MSRs C001\_0044–C001\_0048h) 196

MCi\_STATUS Registers (MSRs 0401h, 0405h, 0409h, 040Dh, 0411h) 196

**Memory-Mapped I/O**

Base 0 Register (Function 1: Offset 80h) 74  
Base 1 Register (Function 1: Offset 88h) 74  
Base 2 Register (Function 1: Offset 90h) 74  
Base 3 Register (Function 1: Offset 98h) 74  
Base 4 Register (Function 1: Offset A0h) 74  
Base 5 Register (Function 1: Offset A8h) 74  
Base 6 Register (Function 1: Offset B0h) 74  
Base 7 Register (Function 1: Offset B8h) 74  
Limit 0 Register (Function 1: Offset 84h) 75  
Limit 1 Register (Function 1: Offset 8Ch) 75  
Limit 2 Register (Function 1: Offset 94h) 75  
Limit 3 Register (Function 1: Offset 9Ch) 75  
Limit 4 Register (Function 1: Offset A4h) 75  
Limit 5 Register (Function 1: Offset ACh) 75  
Limit 6 Register (Function 1: Offset B4h) 75  
Limit 7 Register (Function 1: Offset BCh) 75

**N****NB Machine Check**

Address Register (MSR 0412h) 213  
Control Mask Register (MSR C001\_0048h) 213  
Control Register (MSR 0410h) 213  
Status Register (MSR 0411h) 213  
NB\_CFG Register (MSR C001\_001Fh) 373  
Node ID Register (Function 0: Offset 60h) 46  
Northbridge  
Capabilities Register (Function 3: Offset E8h) 165

**P****PCI I/O**

Base 0 Register (Function 1: Offset C0h) 76  
Base 1 Register (Function 1: Offset C8h) 76  
Base 2 Register (Function 1: Offset D0h) 76  
Base 3 Register (Function 1: Offset D8h) 76  
Limit 0 Register (Function 1: Offset C4h) 77  
Limit 1 Register (Function 1: Offset CCh) 77  
Limit 2 Register (Function 1: Offset D4h) 77  
Limit 3 Register (Function 1: Offset DCh) 77

**Performance**

Monitor 0 Register (Function 3: Offset A0h) 157  
Monitor 1 Register (Function 3: Offset A4h) 157  
Monitor 2 Register (Function 3: Offset A8h) 157  
Monitor 3 Register (Function 3: Offset ACh) 157

**Power Management**

Control High Register (Function 3: Offset 84h) 153  
Control Low Register (Function 3: Offset 80h) 153

**R****Registers**

General MSRs 343  
Model Specific Registers 367  
Power Management 289



**Routing Table**

Node 0 Register (Function 0: Offset 40h) 44–45  
Node 1 Register (Function 0: Offset 44h) 45  
Node 2 Register (Function 0: Offset 48h) 45  
Node 3 Register (Function 0: Offset 4Ch) 45  
Node 4 Register (Function 0: Offset 50h) 45  
Node 5 Register (Function 0: Offset 54h) 45  
Node 6 Register (Function 0: Offset 58h) 45  
Node 7 Register (Function 0: Offset 5Ch) 45

**S**

Scrub Control Register (Function 3: Offset 58h) 137, 140

**SMM**

Save State (Offset FE00–FFFFh) 221

SRI-to-XBAR Buffer Count Register (Function 3: Offset 70h) 143

Status/Command Register (Function 0: Offset 04h) 42

**T**

Thermtrip Status Register (Function 3: Offset E4h) 163

**U**

Unit ID Register (Function 0: Offset 64h) 47

**X**

XBAR-to-SRI Buffer Count Register (Function 3: Offset 74h) 150

